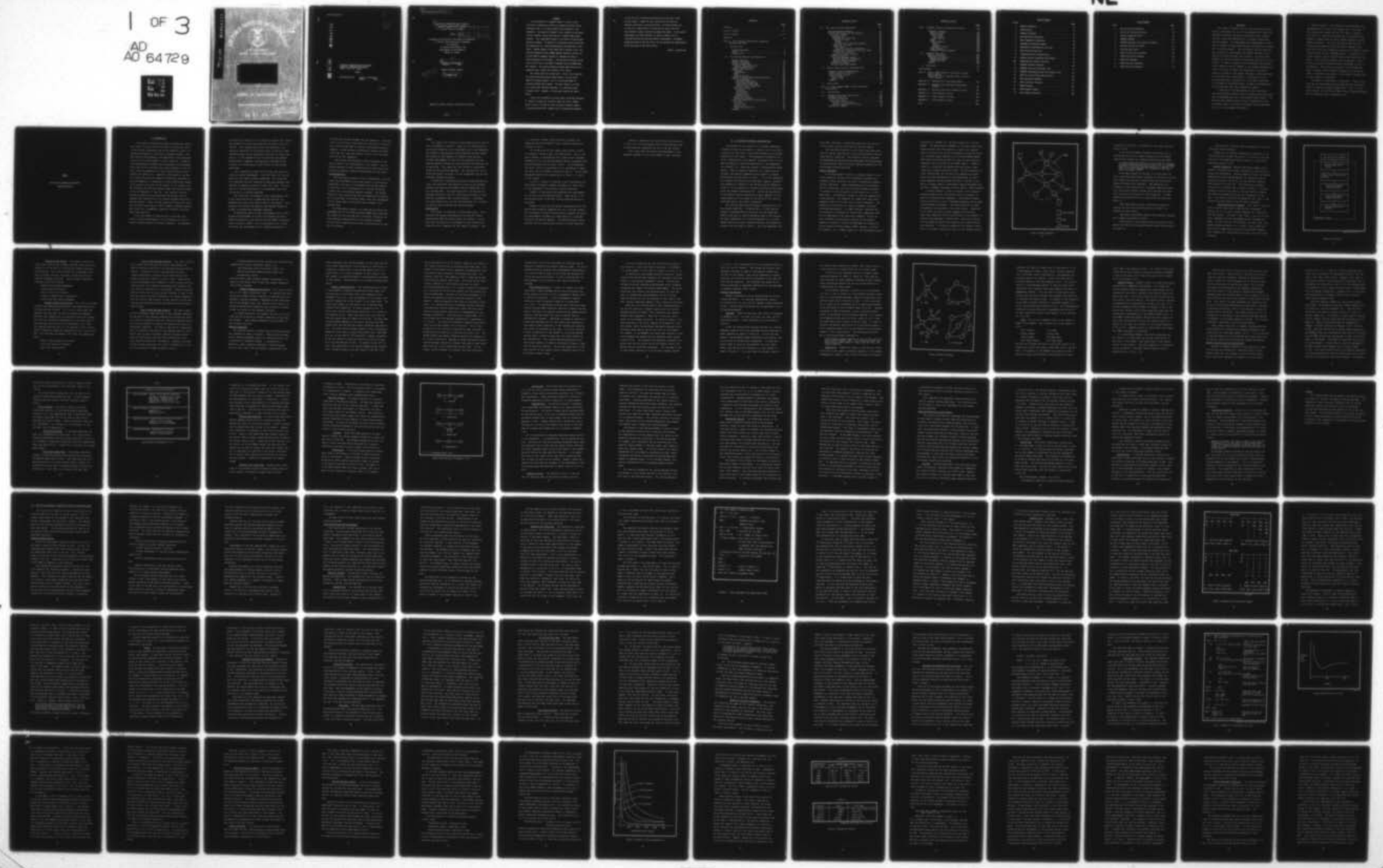
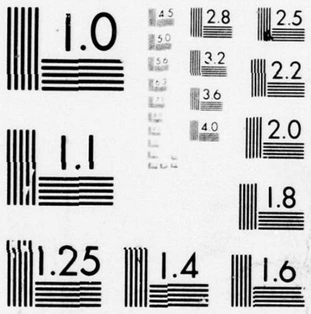
AD-A064 729 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO SCH--ETC F/G 17/2
ELECTRICAL ENGINEERING DIGITAL DESIGN LABORATORY COMMUNICATIONS--ETC(U)
DEC 78 D L RAVENSCROFT
UNCLASSIFIED AFIT/GCS/EE/78-16-PT-1 NL

1 OF 3
AD
A064729




1.0
4.5
5.0
5.6
6.3
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

LEVEL

AIR FORCE INSTITUTE OF TECHNOLOGY

AIR UNIVERSITY
UNITED STATES AIR FORCE

SCHOOL OF ENGINEERING

WRIGHT-PATTERSON AIR FORCE BASE, OHIO

79 01 30 170

AFIT/GCS/EE/78-16

1

ELECTRICAL ENGINEERING DIGITAL DESIGN LABORATORY COMMUNICATIONS NETWORK (Parts 1 and 2 of 3 Parts)

THESIS

AFIT/GCS/EE/78-16

Donald L. Ravenscroft
Captain USAF

(14) AFIT/GCS/EE/78-16 - PT-1, 2

(6) ELECTRICAL ENGINEERING DIGITAL DESIGN LABORATORY COMMUNICATIONS NETWORK. Part 1 and 2.

~~(Parts 1 and 2 of 3 Parts)~~

THESIS (9) Master's thesis,

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

by

(10) Donald L. Ravenscroft
~~Captain USAF~~

Graduate Computer Systems

(12) 247 p.

(11) Dec~~ember~~ 1978

A

Approved for public release; distribution unlimited.

012 225

Preface

The development of a computer network in the Air Force Institute of Technology's Electrical Engineering Digital Design Laboratory (DEL) is a current project being conducted in the laboratory. Two essential elements of this network are the design of both a network routing algorithm and a network node-tc-node protocol. This investigative report is an effort to provide these essential elements. The design of a distributed routing algorithm and suggestions for a node-to-node protocol are described in this report. Another aspect of this report was to develop a data link interface between an Altair 8800b computer located in the DEL and the CDC CYBER 74 computer located in a separate building at Wright-Patterson Air Force Base. This was done to provide the DEL with a direct link to the CYBER 74 computer using an 8080A based DEL computer. The interface computer program description and its supporting User's Manual are included in this report.

This report consists of three parts. Parts 1 and 2 describe the routing algorithm, node-to-node protocol, and the Altair/CYBER 74 interface program. Part 3 is the Altair/CYBER 74 interface program User's Manual. The User's Manual is written as a "stand alone" document therefore, it is published under a separate cover. However, all three parts make up the thesis report.

I would like to express my sincere thanks to the DEL personnel, Mr. Robert G. Durham, Mr. Richard W. Wager, Mr. Dan A. Zambon, and Mr. Orville J. Wright for their excellent technical support in maintaining the Altair computer and its associated peripherals.

I would also like to express my appreciation to Maj Alan A. Ross and Capt Gregg L. Vaughn for their constructive criticism and technical assistance in writing my thesis. My sincere thanks also are due to Ms. Debbi Walters for devoting her time by providing the secretarial support required to prepare the thesis. A very special thanks goes to my Thesis Advisor, Dr. Gary B. Lamont, for his invaluable help during the entire thesis investigation. My deepest appreciation goes to my wife, Carol, for her patience and understanding during the course of the thesis effort.

Donald L. Ravenscroft

Contents

Page

Preface ii

List of Figures vii

List of Tables viii

Abstract ix

Part 1: AFIT Digital Engineering Laboratory Routing Algorithm

I. Introduction 1

Purpose/Approach 3
Scope 4
Organization 4

II. Distributed Network Considerations 7

General Networks 8
Network Hierarchy 12
Network Components 16
Channel Characteristics 17
Node Characteristics 19
Network Architecture 21
Topology 21
Composition 22
Size 24
Network Control 25
General Routing Algorithm Description 26
Control Regimes 28
Isolated Algorithm 28
Distributed Algorithm 28
Centralized Algorithm 30
Deterministic Algorithm 30
Decision Process 31
Isolated 31
Distributed 31
Centralized 33
Deterministic 33
Updating Process 33
Forwarding Process 35
Algorithm Selection Justification 37
Isolated 37
Centralized 38
Deterministic 39
Educational Benefits 40
Summary 41

Contents Cont'd

Page

III. DEL Routing Algorithm Design 42

Background/Requirements 42
Distributed Algorithm Development 45
Decision Process 45
Reachability 45
Reachability Algorithm 47
Response Time 52
Timing 57
Reachability Algorithm Summary 58
Objective Function 59
Hold Down. 59
Objective Function Parameters 63
Reachability Algorithm with Hold Down . . . 65
Algorithm Function 67
Decision Process Summary 72
Updating Process 72
Routing Message Content 73
Routing Message Propagation 82
Update Process Summary 85
Forwarding Process 86
Routing Algorithm Summary 88

IV. Node-to-Node Protocol Considerations 90

General Protocol Requirements 90
Communications Hierarchy 92
Node-to-Node Protocol 94
Level-1 Data 94
Level-0 Data 96
Summary 97

V. Conclusions 98

Part 2: Altair 8800b/CYBER 74 Data Interface Description

VI. Introduction. 102

VII. CYBERSCP Functional Description 105
Software Hierarchy 105
Functional Overview 107
High Level Flow 111
System Initialization 115
Operator Request Processing 117
Functional Summary 118

Contents Cont'd

Page

VIII. CYBERSCP Computer Program Description 120

Computer Programs 120
DOSCYB Program 121
CIN3 Program 121
CIN3 Inputs 123
CIN3 Outputs 128
BASIC 128
CYBER Program 128
Inputs 142
Outputs 142
Program Listing 146
TIME Program 146
CYBERSCP Program Summary 166

IX. CYBERSCP Data Base 167
Memory Partioning 167
Data.Base Parameters 167
DOSCYB Data Base 167
CYBER Data Base 167
Summary 168

X. Conclusion 176

Bibliography 177

Part 3: Altair 8800b/CYBER 74 Interface Program User's Manual (Published under separate cover as Part 3 of 3 Parts)

Appendix A: Reachability Algorithm Example 181

Appendix B: Reachability Algorithm w/Hold Down . . . . Example 192

Appendix C: Routing Simulation Program Listing 200

Appendix D: CIN3 Program Listing 204

Appendix E: CYBER Program Listing 214

Appendix F: Time Program Listing 222

Vita 226

List of Figures

Figure		Page
1	Network Components	10
2	NOS Hierarchy	13
3	Network Topologies	23
4	Routing Control Procedures	32
5	Basic Reachability Algorithm	49
6	Reachability Algorithm Example	54
7	Reachability Algorithm With Hold Down	68
8	Delay Variation With Bias	69
9	Transmission Delay Characteristics	76
10	Updating Process Integrated Flow Diagram	87
10A	Communications Protocol Hierarchy	93
11	CYBERSCP Software Hierarchy	106
12	CYBERSCP High Level Functional Flow	114
13	CYBERSCP Software/data base Initialization Flow	116
14	DATA File Initialization Modifications	124
15	CIN3 Functional Flowchart	130
16	CFILE Functional Flowchart	141
17	CYBER Flowchart	147
18	TIME Program Flowchart	167
19	Altair Memory Partioning	168

List of Tables

Table		Page
1	Routing Algorithm Components	29
2	Typical DEL Transmission Delays	78
3	Typical Propagation Delays	78
4	CYBERSCP Commands	108
5	CYBERSCP Data Base Initialization Parameters	112
6	DOS Modifications for DOSCYB	122
7	CIN3 Data Base Inputs	129
8	CYBER Initialization Parameters	143
9	CYBER System Status Messages	144
10	CYBER Error Messages	145
11	DOSCYB Data Base Parameters	171
12	CYBER Data Base Parameters	173

Abstract

Four types of network routing algorithms were investigated as candidates for use in the proposed AFIT Electrical Engineering Digital Engineering Laboratory (DEL) network. The four types were deterministic, isolated, centralized and distributed. The advantages and disadvantages of each type of network routing algorithm were evaluated for possible use in the DEL network. The distributed routing algorithm was selected for the proposed DEL network because it was found to be more efficient and reliable. The educational benefits of the distributed routing algorithm were also discussed. In order to improve the distributed routing algorithm's response time to changes in the network topology and traffic flow and to reduce the algorithm's oscillation caused by changes in the network topology a technique known as "hold down" was incorporated. A description of the routing algorithm, including this "hold down" technique, is discussed in sufficient detail to permit implementation of the algorithm into the proposed DEL network.

In addition to the description of the distributed routing algorithm, several types of communications protocols were investigated for use in the node-to-node network. The Advanced Data Communications Control Procedures and the Synchronous Data Link Control procedures were recommended for use in the proposed DEL network.

Another subject investigated was the development and implementation of a data link between one of the nodes in the proposed DEL network and the CDC CYBER 74 computer. This data link provides the capability to send data files between a network node (an Altair 8800b computer) and the CYBER 74 computer. The data interface allows the user to selectively manipulate either the Altair computer software using the Altair operating system or the CYBER 74 system software using the CDC INTERCOM system. The selection of either system is easily accomplished using simple user commands. File transfers between the two computers is controlled using the interface software developed in this investigation. The ability to transfer files between the Altair computer and the CYBER 74 computer will allow the CYBER 74 computer to be used as a DEL network resource once the DEL network is developed.

The thesis is organized in three parts. Parts 1 and 2 describe the distributed routing algorithm and the Altair/CYBER 74 interface program respectively. Part 3 is the User's Manual for the Altair/CYBER 74 data interface program and is published under a separate cover.

PART 1

AFIT Digital Engineering Laboratory

Routing Algorithm

I INTRODUCTION

The Electrical Engineering Digital Engineering Laboratory (DEL) at the Air Force Institute of Technology (AFIT) is the primary laboratory used by the students at AFIT for conducting research in the area of computer systems. The DEL contains minicomputers, microprocessors, and integrated circuits to aid the students in their research. Presently the microprocessor systems and minicomputers are autonomous units that do not interface with other computers in the lab. Also, each minicomputer has a very limited number of peripherals available to it. Because of the nature of the DEL, very often, the type and number of peripherals attached to each minicomputer is inadequate or causes additional burdens to be placed on the user of the system. As an example, the DEL currently has one high speed printer that must be shared by all minicomputers in the lab. This sharing is accomplished by physically disconnecting the printer from one computer and reconnecting it to another computer desiring to use the printer. Other limited resources include disk units, user oriented input devices (CRTs), memory, software development tools (e.g. assemblers, compilers, editors), and high speed input devices.

A low cost method of providing all of the DEL with access to existing peripherals is by allowing some kind of resource sharing capability between computers. One method of providing this capability is to design a network that allows the computers to share the resources available in the laboratory. This network would allow the resources to be logically connected to all devices while in reality, these resources would only be physically connected to single computers. In the example of the line printer, the network would permit a computer, not physically attached to the printer, to transfer data to the printer via the network data paths.

This investigative report will discuss two topics related to network development. The first topic will be the design of the network routing algorithm. This algorithm will determine the path required to send data messages from computer to computer using the "least cost" path. The concept of a routing algorithm will be developed along with the design of a routing algorithm.

The second topic discussed in this investigation will be the specification of a communications protocal for messages being sent from node to node in the network. This protocal will specify the node to node message format employed in transferring internodal messages.

An additional topic of this investigation will be the design and development of a data link between the Altair 8800b minicomputer in the DEL and the CDC CYBER 74 computer system located at WPAFB. This investigation will include the design and development of all software necessary to provide data transfer between the two computers. This data link is not directly associated with the design of the network routing algorithms or protocal described previously. However, it does provide an additional resource available to the DEL and, once the network is built, will be accessible by all DEL computers.

This section has provided a brief discussion of the topics that will be investigated in this report. The remaining sections of the Introduction will specifically describe the purpose, scope and organization of the report.

Purpose/Approach

The first major purpose of this investigation is to describe the requirements and design details of the routing algorithm to be used in the proposed distributed network. In addition, this investigation will provide the design criteria for a distributed network protocal. The design details of the routing algorithm and communication protocal will be provided in sufficient detail such that implementation of these items on particular host systems can be accomplished.

The other major purpose is to describe the design and implementation of the CYBER 74/Altair 8800b data interface. The reader will be provided with sufficient information to allow thorough understanding of the computer programs utilized and a complete users manual explaining how to operate the programs.

Scope

This report will provide a brief background and description of networks in general. Based on this description of distributed networks, this investigation will then emphasize the design of the message routing algorithm and the format and content of the computer to computer data protocol. The DEL requirements for each will be presented and the detailed design criteria for the algorithms and protocol will be described. This report will be limited to algorithmic design of the routing algorithm. The algorithm will be in sufficient detail to allow it to be integrated into the proposed DEL network structure.

In addition to providing the network algorithm descriptions, the functional description of each of the software programs that make up the CYBER 74/Altair 8800b data link will be described. This description will include how the programs function as well as a detailed description of the data base parameters used in each program. A comprehensive users manual will also be provided which will instruct the user on how to operate the system programs.

Organization

This report is organized in three major parts. Part 1 describes the design considerations for the DEL network routing algorithm and node-to-node line protocol. Part 2 describes the design and development of the interface between the Altair computer and the CYBER 74 computer. Part

3, is the User's Manual describing the procedures for operating the Altair/CYBER 74 data interface program described in Part 2.

All three parts of this report each contain a table of contents, narrative body, and conclusions. Part 3, the User's Manual, is designed to be a "stand alone" document, therefore it has its own bibliography which is separate from the bibliography used in Parts 1 and 2. The bibliography for Parts 1 and 2 is applicable to all references in these two parts and is located at the end of Part 2. The following paragraphs describe the organization of Parts 1, 2, and 3 of this report.

In Part 1, Chapter II provides a brief background and description of general networks followed by a description of the components of any network routing algorithm.

Chapter III describes the background and requirements for the DEL routing algorithm design. After which, a detailed description of the DEL routing algorithm design is discussed.

Chapter IV describes the design considerations for the node-to-node protocol suggested for use in the DEL network. This discussion includes a description of general the protocol requirements followed by a description of a proposed hierarchy for the DEL network. The detailed design considerations for the node-to-node protocol are then described.

Chapter V summarizes the major design considerations in Part 1 for the distributed routing algorithm and node-to-node protocols suggested for use in the DEL network.

Part 2 describes the design and development of the computer programs in the Altair/CYBER 74 data interface.

II. DISTRIBUTED NETWORK CONSIDERATIONS

The purpose of this section is to provide background information describing networks in general and to provide a rationale for selecting a distributed network as the design objective of this report. The background information contains a description of the general characteristics of a network followed by a detailed description of the components of a network (i.e. nodes and channels interconnecting these nodes). Once the general components of a network are described, the specific components that comprise a network architecture are described in detail. All of the components of a network architecture are described, however, considerable emphasis is devoted to the network control mechanism since it is this aspect of a network architecture that determines any network characteristics. Within the description of the network control mechanism, primary emphasis of this investigation will be placed on the design of the routing algorithm used in the control mechanism.

This section describes the general characteristics of any routing algorithm. Following this, a classification scheme is presented which classifies routing algorithms based on the type of control mechanism used in the network. The basic components of a routing algorithm (i.e. control regime, decision process, updating process, and forwarding process) are described in several. Once the components are described, rationale is presented supporting the choice of a distributed routing algorithm for the DEL network.

The rationale presented in this report describes each type of routing algorithm utilizing the distributed routing algorithm as a baseline. The different routing algorthms are presented by describing the advantages and disadvantages of the particular type of routing algorithm compared to the distributed routing algorithm.

General Networks

In the most general context, a computer network is any inter-connection of two or more computers (hardware and software) and/or terminals and their corresponding communications interfaces. These interconnections can be geographically dispersed, with long distances between computers, or located in close proximaty to each other (e.g. in the same room). The important concept is that the computers be interconnected. The computers can range from large scale computers (IBM 370) to small mini or micro computers. The network can also consist of several computers and their associated peripherals. In some instances, terminals may be connected in a network as "stand alone" components that are not dedicated to any particular computer, but act as user interfaces (input/output) devices to the network. Examples of networks include the Department of Defense Advanced Research Projects Agency (ARPA) network, (1;2;3;4) the Tymshare, Inc., TYMNET system (1), the Telecommunications

Corporation's TELENET (5), the SITA network (6), and the DECNET. The ARPA network (ARPANET) is the most widely documented network and allows users at a particular computer center to access many dissimilar computers around the world. This includes large computers such as the Burroughs B6700, IBM System 370 and the ILLIAC IV as well as many mid-size computers such as the PDP-10 and XEROX Sigma 7. TYMNET is primarily a time-sharing system providing access to many large scale systems around the world. The SITA network provides international air carriers with computerized reservation services worldwide via interconnections to large scale computers located around the world. DEC NET is primarily a network used to interconnect Digital Equipment Corporation's computer equipment (PDP-10, PDP-11, PDP-8 Systems).

A typical network is shown in Figure 1 and includes a backbone network (also referred to just as a network) to which are connected host computers. The backbone network usually consists of a set of message processors called nodes interconnected by channels. The characteristics of nodes and channels are described in the next section. Connected to the nodes are usually other computers called hosts. These host computers utilize the services of the network to transfer data messages to other hosts. Each host contains its own operating system that supports one or more application processes. The primary purpose of the network, therefore, is to permit access by a user, or by a process acting

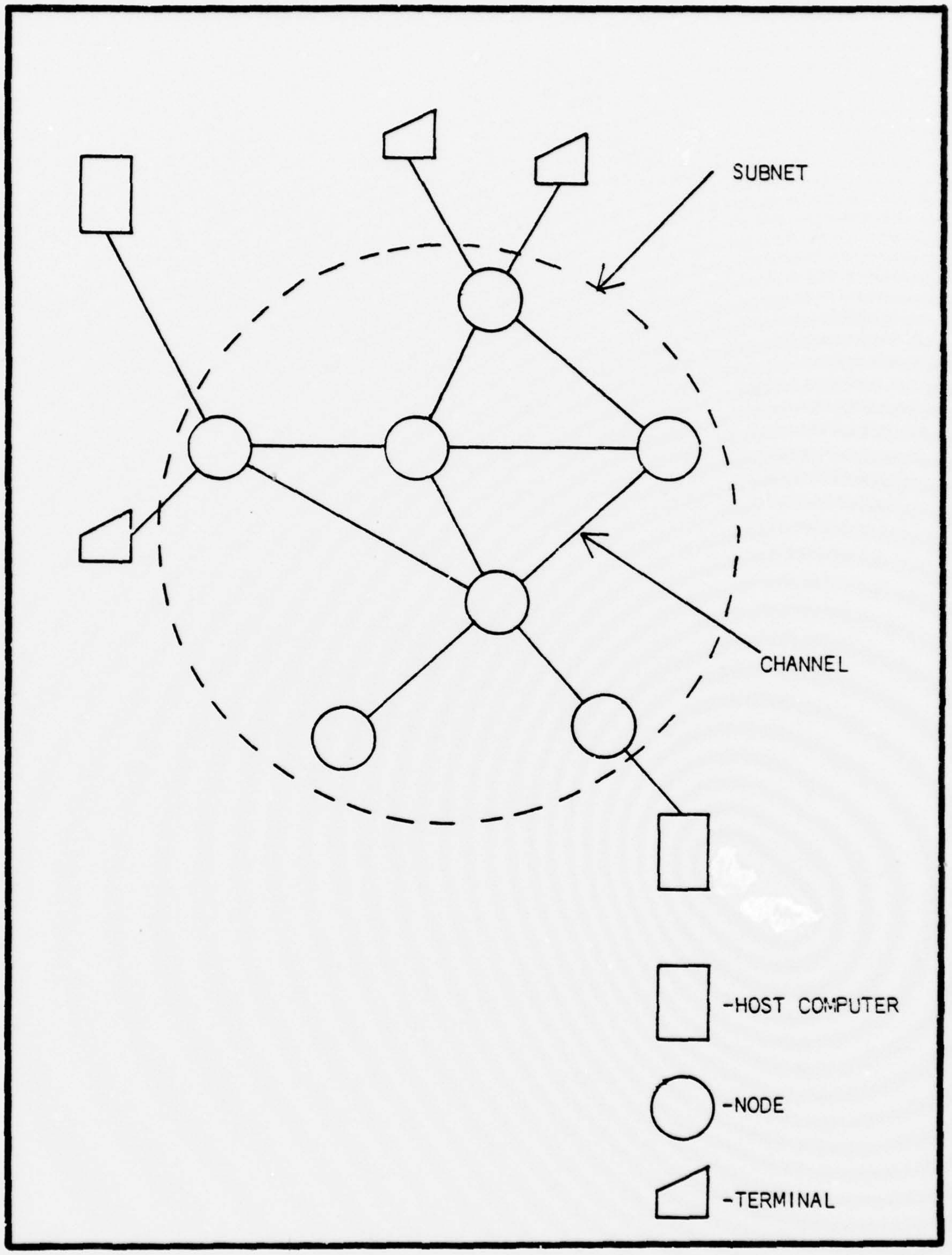


Figure 1 Network Components

on behalf of the user, to resources of the other host computers (7:48).

To utilize the network resources efficiently and to provide the user with a simple interace with the network requires the concept of a network operating system (NOS).

> A network operating system is a collection of software and associated protocals that allow a set of autonomous computers, which are interconnected by a computer network, to be used together in a convenient and cost-effective manner (7:48).

The primary purpose of the NOS is to help users and their associated applications programs to efficiently utilize the hardware and software resources that are distributed among the network host computers. NOS operations are analogous to the operations of a standard operating system on a single-host computer system which provides its users with controlled utilization of local resources. Further operating characteristics of a typical NOS are described below (7):

- NOS removes many logical distinctions between resources that are local and those that are remote (i.e. attached to other hosts).

- NOS makes the backbone network and boundaries between host systems transparent to users.

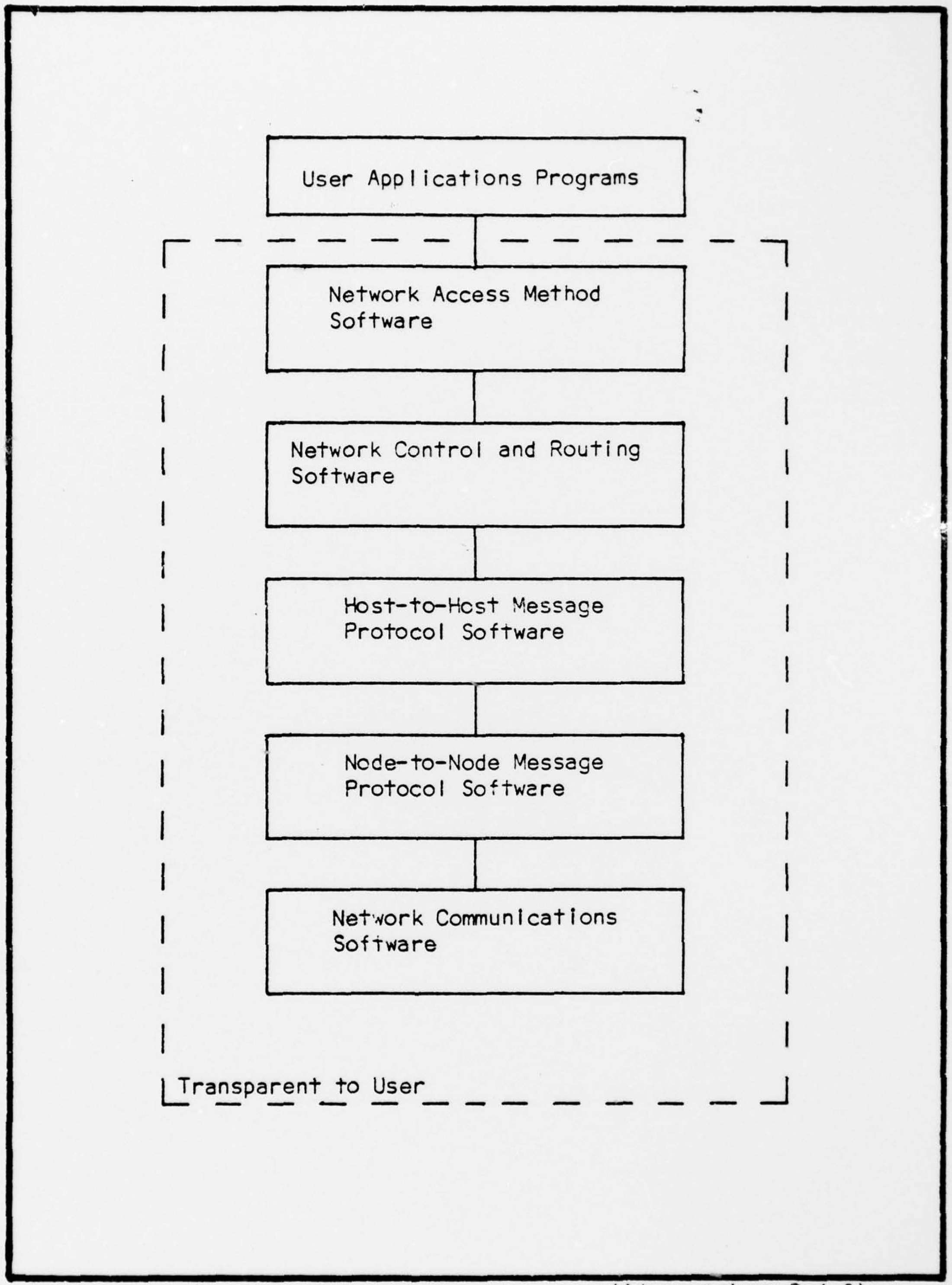
- NOS supports a wide range of easily accessible information services such as system hardware status and resources availability.

- NOS provides relatively easy implementation of distributed data base structures.

Because of the inherent redundancy and autonomy of the constituent host systems, NOS could, in principle, "provide service of far higher availability than can a single-host operating system" (7:49).

Network Hierarchy. Network operating systems usually contain a hierarchical software structure to provide ease of design and maintenance. Typical examples include the Hewlett Packard Distributed System Network (8), Resource Sharing Executive Project (7:49-51;9), and the National Software Works Project (7:51-54; 10). The hierarchical software structure for a typical NOS is shown in Figure 2. The only software visible to the user is the applications software. The remaining software functions are transparent to the user. The specific functions performed by each layer in the hierarchy are described in the following paragraphs.

User Applications Programs. This level is the only level visible to the user or applications programmer. It contains all of the user oriented commands for interfacing with the network. Included in this level is the use of high order languages and user programmed machine language programs. At this level of the hierarchy, the user "sees" the network operating network as a "single" large system with which he can communicate using user oriented NOS commands. These commands are interpreted by the next level of the NOS hierarchy.


User Applications Programs
Network Access Method Software
Network Control and Routing Software
Host-to-Host Message Protocol Software
Node-to-Node Message Protocol Software
Network Communications Software
Transparent to User

(Adapted from Ref 8)

Figure 2 NOS Hierarchy

Network Access Method. The network access hierarchy level contains the software programs that provide the capability to the user of accessing the network functions and resources. This access capability is provided to the user by system intrinsic functions, system services and specific language constructs. Typical network functions provided to the user include:

- Remote host program management.
- Distributed file access.
- Host to host file transfer.
- Access to remote hosts' peripherals.
- Distributed data base management.

Network Control and Routing. This level of the NOS provides the network message control flow mechanism and the routing decision software. It is at this level where the messages sent from host to host are packed and unpacked, buffered, and formatted for transmission within the network. The decisions concerning determination of the best paths to send messages (routing) are also performed at this level. The routing algorithm that will be designed in this investigation, therefore, should be integrated at this level of the NOS. The functions performed at this level are summarized below:

- Node to node message flow control.
- Store and forward of messages.
- Best route determination.

Host to Host Message Protocol. This level contains the software that provides the control and message formatting functions required to send message from one host to another. Information such as message identification, length, destination, and data to be transferred are formatted according to specific protocol rules. These rules allow the messages to be assembled in such a manner that the receiving host can interpret the message and correctly process the text information in the message. Another rule that applies to the host to host protocol software is that it formats the control and data information in a manner compatible with the node to node protocol so that the host to host message may be transmitted within the network using the node to node protocol.

Node to Node Message Protocol. The node to node protocol programs accept the host to host messages formatted by the host to host message software and attaches the necessary control information that will enable the network control and routing programs to transfer the message from source to destination computers. The node to node control information that is attached to the host to host message contains information such as type of message, length of message, source node and destination node, and line synchronization and error detection/correction information. Typically the node to node software generates the control information in a format that conforms to common standards such as:

- Proposed American National Standard for Advanced Data Communication Control Procedures (ADCCP) (11).

- IBM Synchronous Data Link Control (SDLC) (12).

- IBM Bi-Synchrorous Communications (BSC) (13).

- Asynchronous control.

Once the node to node software formats the message it is passed to the lowest level of NOS, the network communications driver software.

Network Communications Driver. This level of software is the lowest hierarchial level. It contains the input and output software drivers required to interface with the hardware interface equipment. Depending on the sophistication of the hardware used, the software drivers may be required to perform error detection and correction functions as well as control of the hardware interface.

As stated previously, the network consists of not only a NOS but also consists of network components called nodes and channels. These network components are described in the following sections.

Network Components

All networks consist of nodes which are interconnected with channels as shown in Figure 1. Nodes serve as concentrators and/or attachment points for external devices such as terminals or computer systems. A concentrator is generally a small computer dedicated to coding messages prior to their entry into the network, combining and buffering messages, and routing messages to their destinations. Since these concentrators involve queuing or buffering, introducing a timing delay in the system, delay time or response time is a critical parameter in their design (1:2). The channel is a physical means of interconnecting the nodes in the network. Characteristics of a channel are described below.

Channel Characteristics. The interconnection of nodes in the network is normally accomplished using copper wire (twisted pair), micro-wave links, high frequency links, telephone lines, etc. For the purposes of this report, the physical composition of the channel is not significant. The significant aspects of the channel are its maximum data transfer rate (capacity), its delay characteristics, its error rate, and any directional constraints (i.e. duplex or half-duplex) (14:7). Data rates used today vary from 100 bits per second (bps) to several million bps. The data rate of each channel depends on the source and receiving node capabilities and the physical channel bandwidth characteristics. The error rates on these channels varies considerably. Generally error rates on voice grade phone lines (1200-9600 bps) are on the order of 10^5 bits/error, however, depending on how the channel is used and controlled, the error rates will vary considerable (15;12). An example is the 10^{12} bit error rate advertised by the Telenet Communications Corporation's TELENET network using data speeds of 56k bps (5:5).

Delay characteristics of the channel depend on the length of the channel and the type of switching algorithms used. The length of the channel varies depending on geographical location and/or type of physical interconnections used. For example, satellite links generally have a longer delay than the relatively shorter land lines. This delay can be significant, especially if half-duplex protocols are used. Switching characteristics (i.e. channel allocation strategies) are a function of not only the channel, but the channel controller mechanisms. The goal of switching strategies is to obtain the maximum utilization from the channels in the network. There are generally two types of switching strategies: (1) Line-switched and (2) message (packet) switched. Line-switching involves dedicating a path from the source node to the destination node prior to commencement of the message transmission. Once the path is established, it is not altered until the message transmission is completed (1: 2; 14:7-10). This kind of switching is generally more efficient and provides better line utilization. This is due primarialy to the relatively long set-up times involved in establishing and physically allocating the paths (1:2). The DATRAN network and common telephone network are examples of this type of switching. Message or packet switching involves routing messages through the network from node to node along paths that are locally determined by each node. In this scheme, shorter messages (or packets) are more efficiently transmitted since various multiplexing techniques may be employed to maximize the channel capacity usage. The use of message switching requires that programmable concentrators (1:2) be utilized as nodes to facilitate the message buffering and routing functions. In this investigation, primary emphasis will be given to this type of switching strategy.

Node Characteristics. A node in a network can range from being a simple hardware device that multiplexes outgoing channels to being a fully programmable computer with a variety of functions. A fully programmable computer acting as a node is generally known as a concentrator (1:2, 14:7). A concentrator receives data from other nodes, buffers the data when necessary, physically repacks the message if necessary into packets satisfying the communications protocol, and then sends this data to another node or attached host computer. Buffering is generally required to ensure that messages are not lost during the transmission from node to node, especially if the incoming and outgoing data rates are different at the node. A critical function of each concentrator in the network is to provide message routing logic if there is more than one output channel at the concentrator. This routing algorithm determines the best output channel to transmit the packet(s) based on some kind of objective function. The routing algorithm is a primary function of the network control mechanism and will be discussed in detail later.

The use of buffering and then transmitting packets is also known as a store and forward technique. The ARPANET is a prime example of this type of network (1:41-57; 2; 3).

In addition to the message switching functions, concentrators also provide data integrity and assurance functions known as "line control procedures" (14:8). The functions include line error checking using hardware and/or software procedures and channel utilization measurement. The concentrator may also contain loop checking logic for ensuring that its neighboring nodes/computers are still active. These functions will not be discussed in this report, however, several references appear in the literature (14;15;16).

Nodes not only provide node to node interconnections, but may also serve as a network access point for various hardware devices (printers, TTYs, terminals) and computer systems (host computers). When this is the case, multi-levels of the network can exist. As shown in Figure 1, the nodes enclosed in the circle compose the backbone or "network proper" while the peripheral equipment attached to the individual nodes may or may not be a part of the network. For example, in the ARPANET only those nodes actually in the circle compose the network and are therefore under network control (14:6). The terminals and computers attached to an access node are not controlled by the network control procedures and are therefore autonomous. This autonomy is one of the primary features of a distributed computer network

(17:13-17). This investigation will be concerned primarily with this kind of network. Even though the terminals and computers attached to nodes in the backbone network are not part of the network, they can be requested as network resources utilizing the control procedures of the node controlling the resources. This provides the capability of resource sharing; an important characteristic and advantage of computer networks (18:9; 19:24).

Network Architecture

Network architecture can be characterized by the following attributes: (1) topology (geographical layout), (2) composition, (3) size, and (4) network control mechanism (14:8). Each of these attributes will be described in the following paragraphs,

Topology. There are basically four kinds of topologies in networks (1:4): (1) star, (2) loop, (3) tree, and (4) distributed (mesh). These topologies are illustrated in Figure 3.

A star (or centralized) topology consists of a central computer system to which are connected various terminals or other computers as shown in Figure 3a. In a centralized network, the peripheral nodes connected to the central site do not serve as concentrators themselves. If these peripheral nodes do serve as concentrators for still other nodes, then the resulting topology is a tree network as shown in Figure 3c. In a star type of topology, there is one central node connected to several other nodes which act as multiplexors or concentrators for still other nodes.

A loop (or ring) topology consists of several nodes connected together as shown in Figure 3b. Each node may have terminals or other computers attached to them, however these attached peripherals do not act as multiplexors or concentrators for other nodes.

A distributed topology is similar to a ring topology (in fact, a ring is subset of a distributed topology) except that in a distributed topology, nodes are connected so that multiple paths exist from node to node. A fully connected set of nodes can be considered to be a distributed topology since multipaths do exist between nodes (14:15-16). However, according to some researchers, a fully connected network cannot be considered as a distributed topology (4:1) because even though multiple paths do exist, nodes can still reach any other node along a direct (trivial) route. In order to preserve the general description of a distributed network, a fully connected network will be considered as a distributed network in this investigation. The resulting definition for a distributed network topology is:

> A distributed network topology is one in which multiple paths exists between nodes. This includes simple networks such as that shown in Figure 3(d) and fully connected networks.

Composition. Composition refers to the physical characteristics of the nodes or attached computers in the network. A homogeneous network is one in which the nodes or attached

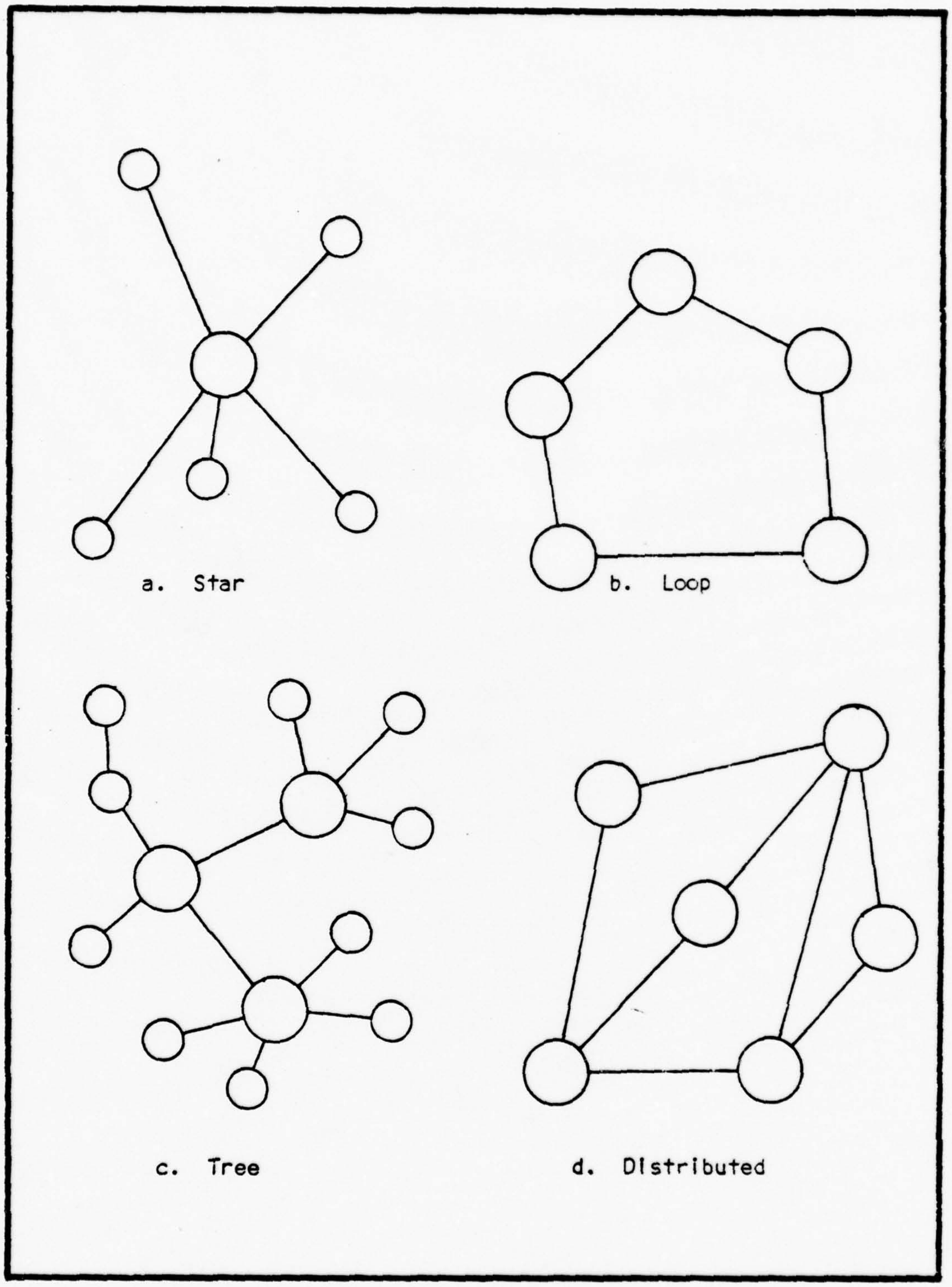


Figure 3 Network Topologies

computers are identical (same types or manufacture) (14:9). In heterogeneous networks nodes and/or attached computers are of different types or manufacture. Heterogeneous has also been applied to networks in which the nodes were of different models of the same computer (14:9). The significance of a heterogeneous network is that the nodal interconnections are in general much more complex than those in a homogeneous network. This can lead to significantly more control and conversion software and hardware being required to interface the network nodes. An example is the connection of non-ASCII standard terminals to the ARPA networks Terminal Interface Message Processors (TIP). This heterogeneous connection required additional programming and the use of at least 1000 10-bit words of main memory in each affected TIP (1:9; 20).

Size. The size of a network refers to the number of nodes in the network. One criteria for sizing networks is (4:2):

small network	1-10 nodes
medium network	11-100 nodes
large network	101-1000 nodes
very large network	over 1000 nodes

The size of a network may have a significant impact on designing the routing and control algorithms used in the network. For example, in the ARPANET each node uses table entries to keep control/network flow parameters of every other node in the network (3:745). This implies that significant modification would be required to expand the ARPANET to 100 or 200 nodes (14:9) from the current size in (1974) of 45 to 50 nodes (2:47).

Network Control. Network control can range from highly centralized to completely distributed. In a highly centraized network (usually topologically centralized, but not necessarily) the control resides in one central node at any particular time. This central node controls the entire network control flow and routing. The central control node may be geographically (topologically) fixed at one node, as in the GE Information Services MARK III network (1:16) or the control may reside in one of many supervisor nodes that are topologically separated as in the TYMNET system (1:26). In the TYMNET system the network contains multiple supervisor nodes that are available to take over network control in minutes in the event of a failure in the current supervisor (1:27; 14:17). Network control is centralized, even though the TYMNET topology is distributed.

In a distributed network, the control mechanism does not reside in a single node, but resides concurrently in each node in the system. Each node shares in the overall control of the network. The prime example of a distributed control mechanism is the ARPA network adaptive control mechanism (1:41; 14:10; 3; 4).

The two main control functions are routing control and data flow control. Routing control consists of the algorithms (or decision processes) required to determine the path(s) (nodes and lines) messages are to follow in transferring from source to destination nodes. This includes initiation and termination of the message transmission. Data flow control is the process of manipulating network messages along the selected paths in such a manner as to provide efficient and error free transfer of data within the network. This control includes procedures for receiving, buffering and retransmitting messages at each node if required. As mentioned previously, this is also known as a store and forward control mechanism. The primary concern of this investigation is the design of the routing algorithm required to implement a distributed network in the DEL. This routing algorithm is described in the following section.

Other control functions in networks include monitoring and maintenance of hardware and software performance, and coordination of these measurements (14:17). This is required to provide adequate failure detection and system repair and to provide a means for evaluating the network performance.

General Routing Algorithm Description

Routing in computer networks is the algorithm or decision process that determines a continuous path of intermediate nodes and lines between a source and destination node, along which messages (or packets) are to be transmitted (1:213; 4:1). Almost all routing algorithms use a table look up procedure for determining the next link and node in the chosen path (4:182). Normally, each node in the network contains routing tables and associated look up procedures for selecting the next link in the message path. An exception to this is when the entire path is preselected prior to message transmission. However, even in this case, the selected path is probably chosen using a directory table look-up scheme (4:129).

There are numerous schemes for classifying routing algorithms. One scheme distinguishes between deterministic and stochastic routing strategies (21). Classification according to the type of control (central or local) is another scheme (22). In addition to these types of routing control schemes, routing algorithms may be adaptive or non-adaptive. Adaptive algorithms adapt or compensate for changes in network topology and message traffic flow patterns. They have the ability to detect system changes and dynamically select the intermediate nodes that will minimize the costs of transmitting the messages to their destination nodes in the network. Non-adaptive algorithms do not react to system changes. Instead they are usually programmed to provide average system response to all system changes (1:214). The routing tables used by non-adaptive algorithms are updated periodically when the source-destination traffic patterns are affected by significant changes. The updated tables are usually generated by a central network control and are then transmitted to the individual nodes in the network.

The classification scheme that will be used in this study was formulated by McQuillan (4). In this scheme a routing algorithm contains the four functions shown in Table 1.

Control Regime. The control regime is the most significant function in classifying network algorithms. Because of this, the nomenclature used in discussing types of network algorithms will be the same as that used to describe the control regime. There are basically four control techniques: (1) isolated, (2) distributed, (3) centralized, and (4) deterministic.

Isolated Algorithms. In isolated algorithms the nodes act independently of each other. All routing decisions are determined using local data exclusively; no explicit information concerning routing decisions are communicated among the nodes (4:173). Figure 4(a) illustrates an isolated node network.

Distributed Algorithms. Distributed algorithms attempt to solve the routing algorithm by sharing routing information among nodes (4:123). Each node uses the information passed to it by other network nodes to update its estimate of the best path to forward messages to particular destinations. Each node in turn passes this best path

Table I

Components of a Routing Algorithm	
Control Regimes:	Governs flow of Routing information. Isolated - independent control nodes Distributed - shared control nodes Centralized - central control Deterministic - fixed control nodes
Decision Process:	Produces Routing choices. Reachability Objective Function
Updating Process:	Updates routing information at nodes. Content of routing message. Propagation of routing message.
Forwarding Process:	Chooses paths for packets. Length of routing directory Width of routing directory

Routing Algorithm Components (4:171)

information to its neighboring nodes. In this manner, the control for the entire network does not reside in any one node, but is "distributed" through all of the network nodes. The ARPA network uses this type of scheme. Distributed algorithm operation is illustrated in Figure 4(b). The primary concern of this investigation is to develop the algorithms required to implement an adaptive-distributed algorithm in the DEL. Justification for selecting this type of algorithm is provided in a later section of this report.

Centralized Algorithm. In centralized algorithms a central network control point (or authority) makes the routing decisions for the entire network. Control information is passed from node to node in the network. However, this information is generated by the central control point (4:173). Nodes in the network may pass network status to the central node to update system routing tables located at the central control point. There may be more than one control point, as in the TYMNET system, but these multiple centers act as back-up systems that take over network control in the event of a failure in the central controller. This is done to help improve the reliability of the centralized network. Figure 4(c) represents this control scheme.

Deterministic Algorithms. Deterministic algorithms do not exchange routing information among nodes and do not attempt to dynamically update their routing information tables. Deterministic algorithms are therefore non-adaptive (4:172). The SITA network (6;23) is an example of a deterministic network. In deterministic algorithms, nodes transfer messages along predetermined lines.

Decision Process. The second function of a routing algorithm is the decision process. The decision process is the basic function of any routing algorithm. It takes the input routing data and chooses (or formulates) the least cost (best) paths for the network traffic. The complexity of the decision process ranges from simple for deterministic network algorithms to very complex for centralized network algorithms. The decision processes for each kind of network routing algorithm presented in the previous section are discussed in the following sections.

Isolated. These algorithms generally use some type of forward and reverse feedback mechanism to choose the best path (4:190-196). The feedback mechanism consists primarily of each node monitoring the behavior of packets received by the node in order to make a routing decision.

Distributed. In the distributed decision process, every node in the network is involved in determining the paths in the network. Nodes exchange routing information periodically which enables each node to adapt to changes in the network topology and traffic flow. An example is in the ARPA network where the nodes exchange "best" estimates of path length with adjacent nodes in the network.

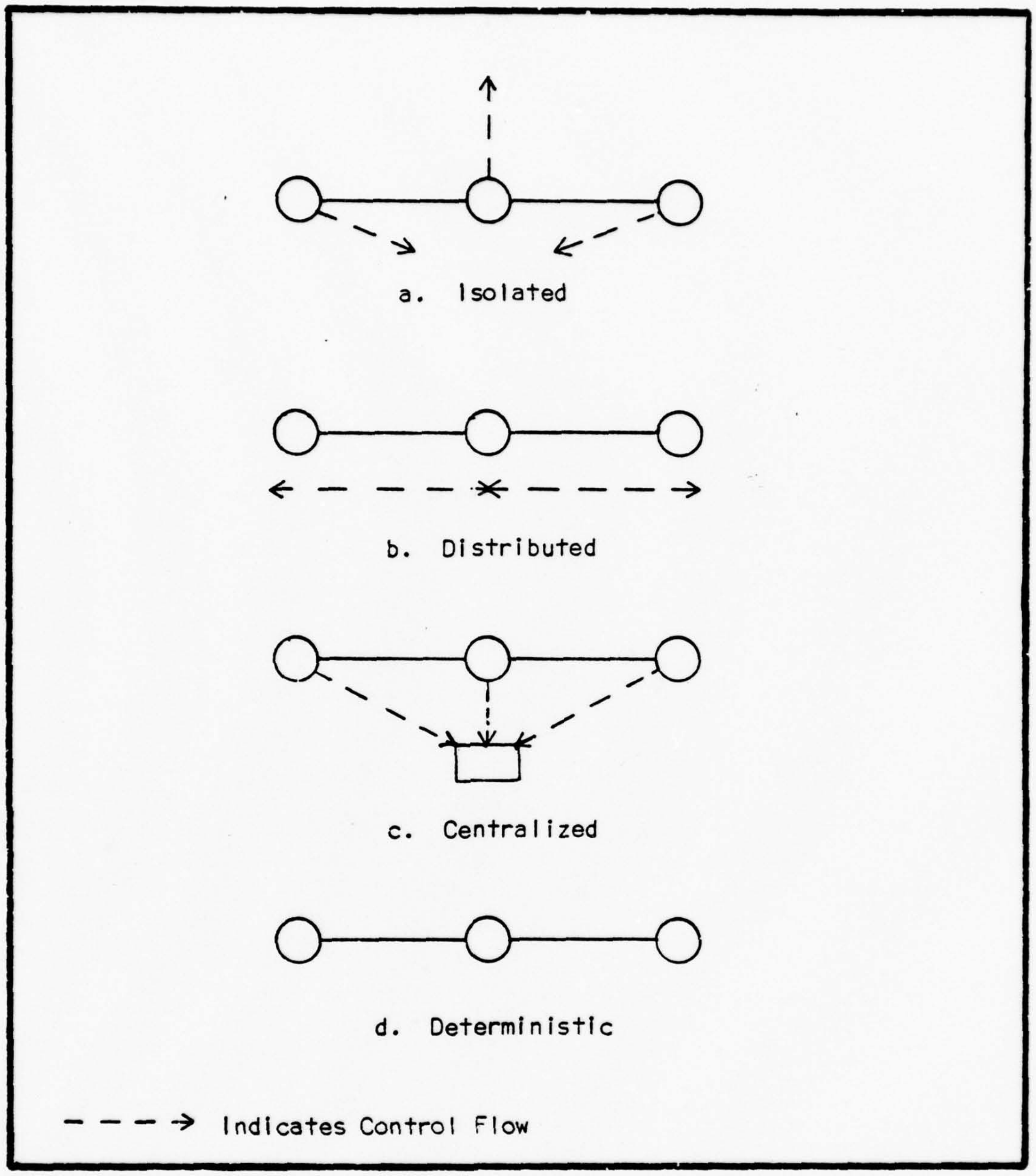


Figure 4 Routing Control Procedures (4:174)

<u>Centralized</u>. Centralized decision procedure usually involves using classical graph theory approaches to attempt to solve global optimization problems (e.g. shortest path algorithms). These techniques generally involve considerable computation effort, especially in large networks.

<u>Deterministic</u>. The decision process in deterministic algorithms is extremely simple since these algorithms are non-adaptive. The choice of paths may be pre-determined by hand periodically, or may use a stochastic method such as assigning probabilities to each output line for forwarding messages (1:216). Another deterministic method may involve flooding all or part of the network with copies of the message to increase the probability of a successful transmission.

All of the decisions processes described above consists of two components: (1) reachability, and (2) objective function. Basically all decision processes determine which nodes are reachable from a given node and assign traffic destined for these reachable nodes to particular paths in the network based on minimizing some objective function. It is beyond the scope of this thesis to go into detail concerning how this is done for each kind of network. However, these components are described and designed in the development of the distributed algorithm described in detail later in this report.

<u>Updating Process</u>. The updating process is the process of inputting data to the decision process and pro-

pagating the results of the decision process to other nodes. Since deterministic algorithms do not perform adaptive functions, the updating process does not apply to deterministic algorithms, but applies only to adaptive algorithms. Isolated algorithms do not require explicit routing information from other nodes therefore, the updating process also does not directly apply to these kind of algorithms. All other algorithms require routing information to be supplied to them, therefore generating the need for an updating process. This process consists of two main parts, the content of the routing message and the propagation method used to forward the routing message.

The content of the routing message is closely related to the objective function used in the decision process. The characteristics of the network that are inputs to the objective function are contained in the routing message. For example, this information may be concerned with link capacity, queue lengths in adjacent nodes, path lengths or delay and response times. In the DEL network the routing information will be primarily concerned with path length information and delays associated with the message queue lengths of adjacent nodes. As in the ARPA network, this routing information will be exchanged between adjacent nodes.

The issue of propagating the routing message through the network is also closely related to the objective function used in the decision process. The routing messages may only need to be sent if changes in the objective function parameters occur or, as in the ARPA network, routing messages may be periodically transmitted by all nodes independently. Whatever method is employed, the progagation scheme should be efficient since routing information is generally sent through the network at relatively high frequencies. In this investigation, the propagation method will parallel that of the ARPA network; that is, the routing messages will be periodically transmitted by each node.

Forwarding Process. The forwarding process is the means used to actually forward each of the messages (packets) along its path from source to destination. As indicated earlier, almost all methods for selecting the next output line from a given node involve some kind of table look-up procedure. Basically, each node contains a routing directory table that correlates destinations with the correct node output line to reach each destination. The node searches this directory for a particular destination and obtains the correct output line. The critical element in the forwarding process is the construction of the directory table; specifically the length and width of the directory.

The length of the directory is the number of possible destination nodes in the network. In a distributed network the routing directory consists of an entry for every other node in the network. Thus each node may route messages to every other node in the network based on the outcome of the decision process. In isolated algorithms the directory may have only one entry that is destination independent. Centralized algorithms generally contain a complete (or near complete) traffic matrix for all nodes in the network. This enables the central control authority to send routing directories to every node in the network with a specific entry for every destination node in the network.

Along with the length of the directory, each directory contains the number of output line choices for each node in the directory. This choice of output lines is called the width of the directory. Each output line entry contains a value used in selecting a particular line to forward each packet. This value is associated closely with the objective function parameters to be minimized in the decision process. In the ARPA network, this value contains the cost (delay and path length) value to reach a particular destination for each output line from the node. Other methods may be used to assign values to the width entries in the directory. For example, the choice of values may be weighted by a feedback mechanism or they may be fixed. Another method is to update the width entry values frequently to provide a kind of load splitting in the network (4:255-257). An important point to consider in designing the width entries in distributed algorithms is that the width entries are updated at a relatively slow rate, therefore, multiple line choices for each destination should be contained in each directory. In the ARPA network and in the DEL network it is desirable to maintain at least two width entries for each destination entry to provide improved reliability in the network.

This completes the components used by McQuillan to classify network routing algorithms. This classification will be used as a basis for development of the network routing algorithm.

Algorithm Selection Justification

This section describes the advantages and disadvantages of each type of network routing algorithm (deterministic, isolated, and centralized) compared against the characteristics of a distributed network routing algorithm. In addition, the educational benefits of the proposed distributed network routing algorithm are described. Based on the comparisons between the different netword routing algorithms and the distributed algorithm and because of the educational benefits associated with a distributed algorithm, the distributed routing algorithm was selected for the DEL network. The following sections describe the relative advantages and disadvantages of each type of network routing algorithm. Following this description, the educational benefits of the distributed network are described.

Isolated. Isolated algorithms, like distribued algorithms, are adaptive, therefore they attempt to compensate for changing network topology and traffic flow. However, the primary problem with isolated algorithms is that they must rely on indirect information about network conditions since each node in the network operates independently without direct transfer of routing information with other nodes. All network information available to the nodes must be gathered from the behavior of the packets flowing through it (4:191). Therefore even using positive and negative feedback techniques, isolated algorithms oscillate (try different paths repeatedly) when attempting to maintain or find the best path through the network. This oscillation occurs even when the network is in a stable state (i.e. there are no changes in network topology or network traffic). There is no solution to the oscillation problem in isolated networks (4:195). Because of this oscillation problem, isolated networks are fundamentally unstable and react much more slowly and inefficiently to changing network conditions than do distributed algorithms.

Centralized. The primary advantage of centralized routing algorithms is that all of the routing logic is done at one location. This implies that the routing algorithm can be less complex to understand and may be implemented using well known techniques. The reachability problem may be solved by finding the transitive closure of a directed graph (4:197). For non-directed graphs Warshall's algorithm (24) may be used. Finding the least path can be solved using Floyds algorithm since the network is centralized (1:201; 25).

The disadvantages, however, are (4:212):

- Considerable computation requiring large processors.

- Processing requirements increase rapidly as the size of the network increases.

- A single control center is unreliable since a single point failure can disrupt the entire system. This implies that backup facilities are not only convienent, but are necessary.

- Updating to adapt to changes in network topology and traffic is inherently slow because traffic must flow to and from a central control center. While the updating process is taking place, the network is using incorrect routing and packets may be trapped for relatively long periods of time between nodes that have been updated and neighboring nodes that have not. Any remoteness of nodes from the central site also introduces significant propagation delays, prolonging the adaptation period.

- Line loading is uneven with heavy overhead traffic occuring near the control center that increases rapidly as the network size increases. Distributed algorithms generally do not suffer from these problems (4:212).

Deterministic. As defined previously, deterministic algorithms include techniques such as fixed routing, flooding and selective flooding, and random techniques (4:189). Deterministic algorithms are unreliable because of the inability of the algorithms to adjust to changes in the network topology or flow characteristics. They are inefficient due to the lack of selectivity in choosing new paths when flooding or random techniques are employed to overcome the poor reliability characteristics of the algorithm (4:186-190). One advantage of deterministic algorithms is that they are relatively simple to design and implement. However, this advantage is diminishing as more information is learned and published concerning the development of distributed techniques.

Educational Benefits. Selection of a distributed network for the DEL will allow a wide variety of current problem areas in network technologies to be investigated in the laboratory. The development of a distributed network will provide a fertile ground for students at the graduate level to participate in the design and implementation of a state-of-the-art field in computer technology because, as stated in a recent technical publication on distributed processing (19:25):

> Numerous problems still face us today in distributed processing...it is too early to tell whether many issues are inherent problems or whether they will be solved or finessed by rapidly evolving technology and understanding.

An additonal benefit provided by the distributed network (note that this is also a benefit of other types of networks) will be the capability to share resources among the minicomputers in the network. This is of particular interest to the DEL because of the limited peripherals now available. This capability to share resources will not only allow higher and more efficient utilization of the DEL facilities, but it may significantly increase the overall computational effectiveness of the laboratory.

Summary

This chapter described the general and specific characteristics of a network and network routing algorithms. The comparative study of several different kinds of routing algorithms lead to the conclusion that a distributed routing algorithm would be most beneficial to the DEL network design. The following chapter will describe the design of this routing algorithm by closely paralleling the description of the components of the general routing algorithm presented in this chapter.

III DIGITAL ENGINEERING LABORATORY ROUTING ALGORITHM DESIGN

This section will describe in detail the design of the routing algoritm to be used in the DEL network. This includes a description of the purpose of the laboratory and the DEL requirements for the overall network. The requirements for the routing algorithm are derived from these network requirements. The routing algorithm will then be described by closely paralleling the routing classification presented previously.

Background/Requirements

The DEL is the primary laboratory facility for all of the digital system oriented courses at AFIT. As such, the laboratory must support many varied projects that utilize the computation facilities in the lab. This includes usage of a variety of mini and micro-computers and peripherals located in the lab. Occasional use of the dial-up data lines to the CYBER 74 computer is also required.

Even though the laboratory contains various mini and micro-computer for laboratory support, a significant short coming is that there are very limited peripherals for each computer. This limitation includes resources such as memory, line printers, high speed storage devices, and high speed input devices for each computer system. This lack of peripherals limits both the utilization of the computers in the lab and also the computational ability of the computers. A method for solving this problem, without a significant monetary cost impact, is to group the computers and existing peripherals into some kind of network with the capability of resource sharing. Designing this network is a complex problem and involves considerable manpower. A significant step in this design process is the design of the message routing algorithm required in the network. This investigation will provide the design of this routing algorithm. This algorithm must be capable of supporting an overall network design that will satisfy the following requirements:

- Provide significant educational opportunities for further research in current network technology
- Provide resource sharing in the DEL
- Provide improvement in the DEL current computational capabilities
- Must be implemented without significant dollar impact
- Must be adjustable to the DEL physical layout.

The selection of a distributed routing algorithm should support all of these network requirements.

The distributed routing algorithm that will be described in this investigation is only part of the overall network operating system (NOS) discussed previously. Significant areas of the NOS must still be required to be developed. These areas will provide a variety of educational opportunities for further network design and investigation.

The areas remaining to be developed include a network flow control mechanism, host to node protocols, host to host protocols and low level communications software and hardware drivers.

Resource sharing is a function of the overall network operating system; the routing algorithm supports the concept of resource sharing by providing the NOS with the capability of directing messages reliably throughout the network. This routing will allow users to communicate, and therefore share resources, with other computers in the network.

Improvements in the DEL computational capability is an significant benefit of the DEL network. By allowing resource sharing and essentially distributed processing, the laboratory will be able to more efficiently utilize all of the computers in the laboratory as well as all of the diverse peripherals.

Because the network will require relatively few new purchased hardware or software modules and will utilize existing facilities in the DEL, the cost in dollars of developing the network will be relatively small. Significant student manpower will, however, be required to implement the complete network.

Perhaps one of the routing algorithm's strongest points is its ability to accomodate any physical interconnection of computers with relative ease. Because of this, the computers in the laboratory can be easily interconnected into a network without requiring any physical rearrangement.

The next section of this report describes the selected routing algorithm.

Distributed Algorithm Development

This section will discuss the design of the routing algorithm. The algorithm development will follow the same general categories used previously in classifying routing algorithms. Each component of the routing algorithm listed in Table I will be described in detail in the following sections. The control regime selected is the distributed control technique described previously. This distributed control is localized at each node, therefore, each node in the network will be required to execute the routing algorithm. Since the control regime component has already been selected, the following sections will describe the three remaining components of the routing algorithm (i.e. the decision process, the updating process, and the forwarding process).

Decision Process.

The decision process contains two design considerations: (1) reachability, and (2) objective function. Each of these design considerations will be described for the DEL routing algorithm.

Reachability.

The reachability of the DEL algorithm is concerned primarily with determining the best path from a particular node to all other nodes in the network. The selection of this path depends on the characteristics of the objective function. For the description of the reachability portion of the routing algorithm, "best path" will mean the minimization of the objective function that will be discussed in the next section. Understanding of the design of the reachability portion of the routing algorithm does not necessitate a detailed understanding of the objective function. The significant point is that the objective function will be a minimization process.

Before proceeding with the description of the reachability algorithm, a brief review of the physical properties of a node are required. A node consists of the hardware and software necessary to support the distributed network architecture. The hardware consists of standard logic processing elements (e.g. CPU and memory) and communications interfacing elements. The constraint is that there exist at least two input/output lines from each node for reliability. The software must, of course, contain the network control and routing algorithms along with sufficient buffer storage to permit efficient control flow through the network.

The objectives of the reachability portion of the routing algorithm are: (1) to provide the node NOS software sufficient routing information (i.e. output line number) for transmitting a message to a particular destination and, (2) to provide the capability to change this output line to adapt to changes in the network topology or traffic flow.

The two objectives are closely related since selection of the proper output line depends on the topology and traffic flow of the network. These two objectives, then, will be described concurrently in the description of the reachability portion of the routing algorithm.

Reachability Algorithm. The reachability algorithm is table driven. All of the information required to arrive at the proper output line decision is stored in tables (or arrays) in the node's memory. The node keeps a table, C, which has an entry per destination containing the minimum cost to reach all other nodes (destinations) in the network. If the cost to any destination node exceeds the maximum allowable cost than the value MAX is put into the table entry for that destination. Cost in this description is a function of the objective function and typically implies distance or delay to a particular destination. The cost of reaching itself is defined to be zero. The cost to send a message out on a particular line (i.e. to an adjacent node) is stored in the A table. This table therefore has NL entries, where NL is the number of output lines for the node. Each node also contains a neighbors' cost array, NC, which contains the neighbors' best estimates to reach any other node in the network. The NC table has one row per destination and one column per output line. Each node periodically sends its minimum cost matrix, C, to its adjacent nodes where it is copied into the ℓth column of the neighbors' cost array, NC.

ℓ is the line number on which the C matrix was received by any particular node.

The directory or routing matrix, R, contains the output line number representing the mode's best path to any destination.

The algorithm continually receives its adjacent nodes' best path estimate and places this estimate in its NC matrix. It then uses this received information to estimate the cost to each destination node in the network which it stores in its C matrix. The node places the output line number corresponding to this best path into the R matrix for each desination node. Finally the node sends the new C matrix, containing its most current best path information, to all of its neighboring nodes. This algorithm is shown in Figure 5.

The first step in the algorithm is to copy the recieved minimum cost table, C', into the NC table. This is done by copying the C' table into the ℓth column of the NC array, where ℓ represents the line number to the adjacent node. Steps 3-7 are run after the NC table is updated. While steps 3-7 are running the NC table cannot be updated otherwise the C table may not contain correct information. However, updating the NC table while steps 3-7 are running, will not introduce erroneous information into the network for longer than one computation of steps 3-7. No significant incorrect information will be introduced into the network. This timing is discussed later in this section.

```
1.  ℓ = line number to adjacent node
2.  FOR i = 1 to N do
    NC(i,ℓ) = C'(i)   ; where C' = received
    Next i              minimum cost matrix from
                        node on line ℓ
3.  FOR i = 1 to N do; i = destination counter
    C(i) = MAX        ; set cost to i to MAX
    FOR j=1 to NL     ; NL = number of output lines
     IF NC(i,j)+A(j) ≥ C(i) then go to step 4
                      ; if the total cost to destination i
                        is greater than the present cost
                        then continue with next i
     C(i)=NC(i,j)+C(j);if not then update the new cost to i
     R(i) = j          ; store the line number
4.  Next i
5.  Next j
6.  C(self) = 0        ; cost to itself = 0
7.  R(self) = 0        ; cannot reach itself
8.  send out C table to adjacent nodes
```

Figure 5   Basic Reachability Algorithm (4:203)

Step 3 is the main part of the reachability algorithm. In the outer loop (indicated by the i index) the index i points to a destination node i. The outer loop therefore loops through all of the N destinations in the network. Since the cost matrix, C, is being updated to contain the best path estimate to each destination, i, the algorithm must first assume that the destination, i, is not reachable therefore C(i) is assigned the value, MAX.

In the inner loop (j index loop) the cost to reach the i^{th} destination using the j^{th} output line is calculated and compared with the cost perviously stored in C(i). If the new cost is less than the old cost, then the new cost is stored in C(i) and the output line number, j, is stored in the reachability matrix R for the i^{th} destination. In essence then, C(i) is updated with the least (or best) cost to reach any destination and the output line number to reach the i^{th} destination is stored in R(i). The flow control mechanism can therefore use the R(i) value to determine on which line to send any message that is to be sent to the i^{th} destination. It is significant to note that this algorithm does not depend on prior knowledge of the network topology. It only requires the objective function parameters to determine the best path to the final destination. In the algorithm shown in Figure 5, the objective function parameters are the adjacent nodes' best estimate to reach destination i, and the node's cost to transmit a message out of line j. These two parameters are summed producing the nodes current estimate to reach destination i using output line j. The calculation of these parameters will be discussed in detail later in this report.

After looping through all of the destinations (i.e. i > N) the node must set the cost to reach itself and the reachability to itself to zero. The C matrix is now completely updated and can be sent to all of the adjacent nodes using the node's flow control procedures. An example of the routing algorithm is described in Appendix A.

The first objective of the routing algorithm, providing the correct output line, is satisfied by calculating the best path and storing this data in the reachability matrix, R. The second objective of the routing algorithm, adaptability to network changes, is inherent in the algorithm itself. The algorithm constantly receives updates from its adjacent nodes and uses these updates in its objective function calculations In this manner, the node can sense changes in the network and then update its best cost estimate, C(i) and reachability matrix R. However, since the algorithm receives and passes network information only between its adjacent nodes, it takes a finite amount of time to propagate network changes completely throughout the entire network. This finite amount of time is called the network response time. Since the time to respond to changes in the network is a measure of the routing algorithm's efficiency, it is desirable to reduce the response time. Therefore, reduction of the routing algorithms response time is an important consideration of the reachability algorithm.

<u>Response Time</u>. Response time is the finite time between when a change in the network occurs and when the routing algorithm adapts to this change (4:220). This time depends on the topology of the network and the method in which the reachability algorithm adapts to the change in the network. Clearly the characteristics of the topology of the network (i.e. size of the network and the interconnectivity of the nodes) will cause the response time to vary. More important, and of primary concern, is the manner in which the reachability algorithm adapts to the changes in the network. For the purposes of the present discussion, the objective function used in the algorithm shown in Figure 5 will minimize only the number of hops (one node per hop) to go from the source node to the destination node. Each hop will have a cost of unity associated with it. In essence then, the objective function will find the shortest path from source to destination node. The section describing the objective function will expand the objective function to also include the least delay path as well as the shortest path. To understand the problem associated with the response time of the reachability algorithm, the mechanics of the algorithm in an adaptive environment must be considered.

Since each hop has a cost of unity associated with it, the hop count to a given node increases smoothly as the distance to that node increases. Furthermore, no node ever has a hop count to a given destination node that differs by more than one count from any of its neighboring nodes. If the number of hops to a given destination gets better (decreases) then because of the design of the algorithm, the nodes will all agree on the new better value in a very short time. If the hop count to a given node gets worse (increases) the nodes will not accept the higher count while they still have adjacent nodes that have the old lower count. In other words "the reachability algorithm reacts very quickly to good news and very slowly to bad news" (4:217). A simple case suggested by McQuillan (4) illustrates this situation. Figure 6 shows how nodes one through four react to node zero coming up and going down in the best and worst cases. The example introduces an important point, that is, the order in which the nodes calculate and propagate routing information has a significant impact on the time it takes to respond to a given topological change. For the purposes of this illustration, we will assume a definite order exists for computing and exchanging routing information among the nodes. In reality, the order is a function of many variables and changes dynamically as workloads vary within each node. The important point is that the response time is affected by the order in which the nodes exchange routing information in response to a given failure. (McQuillan provides a much more intensive study of the efficiencies of reachability algorithm propagation methods in his dissertation (4:405-431)). Figure 6(a) shows the results when node zero comes

GOOD NEWS

```
0-----1-----2-----3-----4          0-----1-----2-----3-----4

DN   MAX   MAX   MAX   MAX         DN   MAX   MAX   MAX   MAX

UP    1     2     3     4          UP    1    MAX   MAX   MAX

                                         1     2    MAX   MAX

                                         1     2     3    MAX

                                         1     2     3     4
```

a. Left-to-right sequence, test case, node 0 come up

b. Right-to-left sequence, worst case, node 0 comes up

BAD NEWS

```
0-----1-----2-----3-----4          0-----1-----2-----3-----4

UP    1     2     3     4          UP    1     2     3     4

DN    3     4     5     6          DN    3     2     3     4

      5     6     7     8                5     4     3     4

         ...                             7     6     5     4

                                         9     8     7     6

     MAX   MAX   MAX   MAX              11    10     9     8

                                                ...

                                        MAX   MAX   MAX   MAX
```

c. Left-to-right sequence, test case, node 0 goes down

d. Right-to-left sequence, worst case, node 0 goes down

(4:218)

Figure 6 Reachability Algorithm Example

up. Initially all of the nodes (one through four) are indicating node zero as unreachable (cost is MAX). As node zero comes up, node one accepts the new cost (one) immediately. Likewise, nodes two through four immediately accept the new values passed to them from each node from the left. Figure 6(b) shows the same situation but the updating occurs from right to left. Notice that it takes node four considerably longer to obtain the new information since it must wait until its neighbor, node three, calculates and passes the new better routing information. Node three must wait for two, etc. In Figures 6 (c and d) the example network is shown reacting to bad news (node zero going down). Note that in each case each node, beginning with node one, refuses to accept that it cannot reach node zero until it has exhausted trying its alternate path through its other neighbor. This search for the new best path, once the old best path gets worse, is known as rise time. Correspondingly, the time, illustrated in Figure 6 (a and b), needed to adapt to a better path to a given destination is called the fall time (4:219). From an optimization standpoint, it is clearly desirable to minimize both rise and fall times in routing algorithms.

The reachability algorithm in Figure 5 behaves in exactly this manner. The fall time of the algorithm is short because as soon as it receives information about a better path to destination i, it will update the value in C(i) and R(i) to reflect this better path. This is also known as a zero fall time. The rise time, however, is considerably longer. In order to find a better path once the previous best path gets worse, the algorithm must search for successively longer paths until a better path is found or no paths exist that satisfy the MAX cost constraint. A significant point to consider is that while the algorithm is searching for a new best path, the node must "believe" that the destination node is still reachable, therefore, it must continue to route information to it (4:227). The problem then becomes what to use as the best path while the search for the new path is in progress. The solution to this problem is suggested by McQuillan (4:228) and is called "hold down". The example shown in Figure 6 was concerned with the shortest path solution from each node. However, the algorithm in Figure 7 is a more general algorithm which can also satisfy the problem of finding the best path using minimum delay between source and destination nodes. This more general algorithm has the same problems as the shortest path example and therefore the hold down solution applies to it as well. A detailed discussion and the algorithm used to implement hold down is described in the objective function section later in this investigation. Basically, however, the hold down solution is:

> the routing algorithm should continue to use the [old] best route to a given destination, both for updating and forwarding [messages], for some time period after it gets worse (4:229).

Hold down, therefore, assumes that until better information is present or the determination is made that the destination is unreachable, the node should continue to use the old best path to forward network messages.

Another important aspect of the reachability algorithm is the subject of timing (i.e. when should the reachability algorithm be activated).

Timing. The activation (running) and deactivation of the reachability algorithm is controlled by the overall NOS program. A significant timing constraint should be considered when designing the activation software. The values in the C matrix are a function of the values in the NC matrix and the A matrix. Since the NC and A tables could possibly be updated while the C matrix is being updated, a classical synchronization problem (26:68) exists. For example, if the NC matrix for destination i is updated while the reachability algorithm is processing the i^{th} destination, then the possibility exists that the cost to reach this destination, C(i), may be incorrect. This will happen if the algorithm is still searching for the best output line (i.e. $1 < j < NL$) and the NC matrix is updated with a better path estimate for an output line, say x, that the reachability algorithm has already searched (i.e. $1 \leq X < j < N$). The value in C(i) and therefore R(i) will be incorrect. The same situation also applies to the A matrix. The constraint, then, placed on the NOS is that while the i^{th} destination is being updated by the reachability algorithm, the NOS should prevent the i^{th} destination information in the NC array or the A array from being updated. The consequences of allowing the NC and A tables to be updated without observing this constraint, are not severe on the system. In the worst case, the values in C(i) and R(i) will be incorrect for one complete execution of the reachability algorithm. The next pass through the algorithm will use the new updated information. Another solution to this synchronization problem is presented in the objective function section of this report.

<u>Reachability Algorithm Summary</u>. Because of the manner in which the algorithm receives network information and calculates the best path to any destination based on this information, the algorithm can successfully satisfy the two objectives described in the beginning of this section. The selection of the best path to any destination, the first objective, is accomplished by minimizing the objective function. The algorithm adapts to changes in the network, the second objective, by using information from its adjacent nodes and in turn passing its estimate of the best path to its adjacent nodes.

Associated with the reachability algorithm, however, are timing constraints that must be observed by the NOS implementing the reachability algorithm. By controlling the synchronization involved in changing the objective function parameters, the NOS can prevent the reachability algorithm from outputting erroneous information. Another significant problem associated with the reachability algorithm is that of improving the rise time it takes for the network to adapt to bad news in the network. The introduction of the "hold down" technique to the original reachability algorithm reduces the rise time such that the algorithm more efficiently reacts to bad news in the network.

Since the hold down technique is related closely to the objective function, the next section of this report will describe the hold down technique along with the objective function parameters.

Objective Function. This section will continue to build on the concepts introduced in the reachability algorithm description of the previous section. First, since understanding the hold down technique mentioned in the previous section is necessary to understanding the final reachability algorithm, the hold down solution to improving the response time of the overall routing algorithm will be described. Next the parameters of the objective function to be used in the DEL routing algorithm will be described. Hold down will then be applied to these parameters to produce the final DEL reachability algorithm that will be suggested for use in the DEL routing algorithm.

Hold Down. The hold down technique that will be described was first discussed by McQuillan (4). The description that will be presented in the following paragraphs is a general description of hold down as it applies to the "best path" selecting process. The term "best path" in this description refers to any choice of objective function parameters (i.e. distance, delay, throughput, capacity, etc.). Once a general description has been provided, the specific application of the hold down solution to a specific set of objective function parameters will be described.

As mentioned in the previous section, hold down is concerned with the problem of what to use as the best output line to a given destination while the reachability algorithm is searching for a new best path. Without hold down, once the reachability algorithm begins its search for a new best route, it will cause an oscillation in the network. This oscillation is caused because the cost, C(i), is changing (i.e. taking on better and better values as they are found) as the search progresses. The algorithm, without hold down, will generate each new value in C(i) as a new best path to i, therefore, until the true best path to destination i is found (assuming one exists), the network will oscillate. Hold down will prevent the oscillation from occuring. Hold down then, as stated earlier, implies that the routing algorithm should continue using the previous best path for some time period after the path gets worse. The reachability algorithm should continue to report the current value of the previous best route to its neighbors and should continue to use this best route for routing packets for some given interval of time (4:229). Two main issues that result from using hold down are: (1)

when should the reachability algorithm enter hold down and (2) how long should the node remain in hold down.

Initiating Hold Down. The node should begin hold down when the cost of the best line gets worse. Also, if the line (or node attached to this line) goes down and this line is the best line to a given destination, then enter hold down. The purpose for entering hold down in these cases is to purge the system of the old routing information so that the new routing information will be believed. A convenient way of detecting when to enter hold down due to a node failure is for the node going down to transmit its cost matrix, C(i) with the value MAX for all destinations, i, to each adjacent node. In this manner, the node receiving this going down C matrix, will enter hold down for just those nodes for which the dying node (line) was the best path. If the node that is going down is unable to send the C matrix before the line fails, then the line failure will eventually be detected by the adjacent nodes by using some kind of line failure detection mechanism (16:22; 27:555). In any event, the node must enter hold as indicated above when the cost of the best line gets worse. An important consideration once the node enters hold down is how long to remain in hold down.

Hold Down Duration. The duration of hold down is measured using a counter. There must be a counter for each destination, i, along with the corresponding identity of the line, R(i) and cost, C(i) for the best path to i. The counter for each destination keeps track of the metric used to measure the hold down duration for each destination node. Several points should be considered in order to use this counter (4:232-236):

(1) If the best line gets worse and then better again, the counter should be allowed to run to completion. This is done to ensure that spurious routing information is not injected into the system by the line changing its value. (See McQuillan for a detailed example of this phenomenom).

(2) The hold down counter must be long enough for the new routing information to propagate from the node in question to all adjacent nodes and back again (4:234). This is done so that all old routing information concerning the old best path is purged from the network thus preventing this old information from inpeding the adaptation to the new best path information. Generally the hold down counter need only wait long enough for the new information to pass to and from the immediately adjacent nodes. In some cases, however, this may not suffice. In such cases, hold down must last long enough for the new information to propagate through several nodes (4:234). "The exact number of nodes [hops] depends on the likelihood of a path of several hops being superior to one of the fewer hops"(4:235). Since the cost to reach any particular destination from a given node differs by only one from the values held by any adjacent node when using the shortest path objective function, hold down need last only long enough for the new routing informa-

tion to propagate to the adjacent nodes. If delay is being minimized then the counter must be longer and is a function of the delay variables. However,

> "the penalty for a hold down counter [timer] which is too short is not severe; routing will oscillate when the recently-acquired alternate route is found to be no better after all" (4:235).

An example of this oscillation is found in McQuillan (4:234-235).

(3) The hold down counter must be inversely proportional to the routing propagation frequency. If routing information is sent less frequently or there are long delays on the line then the counter should be longer.

The exact values measured by the counters, as suggested above, are functions of the characteristics of the network (line baud rate, line capacity, topology, etc.) and the type of objective function chosen for the reachability algorithm. The next section describes how the hold down counter is implemented for particular objective function parameters that are suggested for use in the DEL network.

<u>Objective Function Parameters</u>. This section will describe two types of cost function parameters, path length and total delay, that can be used in the objective function of the reachability algorithm. The final reachability algorithm will then be described using these parameters and incorporating the hold down counter described in the previous section.

The first cost parameter is the distance to reach a particular destination. This distance is measured by the number of hops or the number of nodes required to go from a given node to any destination in the network. Generally, this cost parameter is minimized by finding the shortest path from the node to a particular destination.

The second parameter is the total delay seen along the path from a given node to any destination node. This delay can be a function of many parameters such as length of the path, path propagation delay, line capacity in bits per second, and delay through intermediate nodes (1:215). Generally the delay parameter is some quantitative measure that is an estimate of the time it takes a message to flow out of any given line from a node to a given destination. A detailed discussion of estimating delay can be found in McQuillan (4:86) and Schwartz (1:61). One such measurement parameter is the estimate of the queueing delay for the output buffer associated with a given output line to a particular destination. The actual quantitative value of this delay parameter is a function of the type flow control mechanism implemented in the node and is beyond the scope of this report. However techniques for estimating this delay are found in Schwartz (1:221). The important point to consider in this investigation is that the delay parameter used in the objective function is a meaurement (or estimate) of the total delay to send a message from a given node to a particular destination using the node's selected output line. The objective function minimizes this parameter in order to select the best path. It is worth noting at this point that the shortest path parameter described previously can be considered as a specific case of the total delay problem. This is achieved by assuming that the delay contributed by each intermediate node is equal to one.

Now that the parameters that compromise the objective function have been described, the DEL reachability algorithm using these objective function parameters, in conjunction with the hold down technique described earlier, will be described.

Reachability Algorithm with Hold Down. This section will describe the objective function used in the DEL reachability algorithm and the addition of the hold down logic to the previous algorithm shown in Figure 5. The resulting improved reachability algorithm with hold down is shown in Figure 7.

Since one of the primary purposes of the routing algorithm being developed in this investigation is to support an overall network structure that will provide educational opportunities to AFIT students, the selection of cost parameters for the objective function must be general in nature. In other words, the cost parameters must provide the capability to experiment with the quantitative values of the objective function in order to investigate various aspects of network control and routing mechanics. This implies that the objective function must be general enough to allow for various cost parameters to be studied, while at the same time still provide a solution to the reachability algorithm.

In order to satisfy this generality constraint, the objective function selected for the DEL reachability algorithm will minimize the total delay from a given node to any particular destination out of the node's corresponding output line. The objective function can be written as:

$$C(i)\min = \min_{\ell}(NC(i,\ell)+A(\ell)+\beta)$$

where: $1 \leq \ell \leq NL$, NL = number of output lines

$1 \leq i \leq N$, N = number of destinations

The minimum cost to destination i, is the minimum sum of the neighbors' estimates to reach destination i, the nodes estimated delay out of line ℓ, and a bias term , β.

The neighbors estimate, C' in Figure 7, is received from each adjacent node the same as it was in the previous algorithm (Figure 5). The $A(\ell)$ matrix element represents the node's cost to transmit a message out of line ℓ. As suggested by Schwartz, this could be a direct measurement or estimate of the queueing delay for the specific output line, ℓ (1:218-220). The bias term, β, is added to the objective function to reduce looping or "ping-pong" effects in which messages return to a node from where they were previously transmitted (1:219-220). Addition of the bias term thus helps improve the algorithm's effectiveness. The technique of using this bias term is called the "shortest time plus bias" routing algorithm (1:220). The optimum value of β is normally found through simulation studies by varying the size of β and observing the resulting packet delay. A typical simulation curve showing the average time delay of a single packet in a distributed network is shown in Figure 8 (1:221; 21). From this curve, the typical value of β is about 10 msec.

The algorithm shown in Figure 7 includes the hold down technique described earlier. The following discussion will describe how the algorithm functions using this technique.

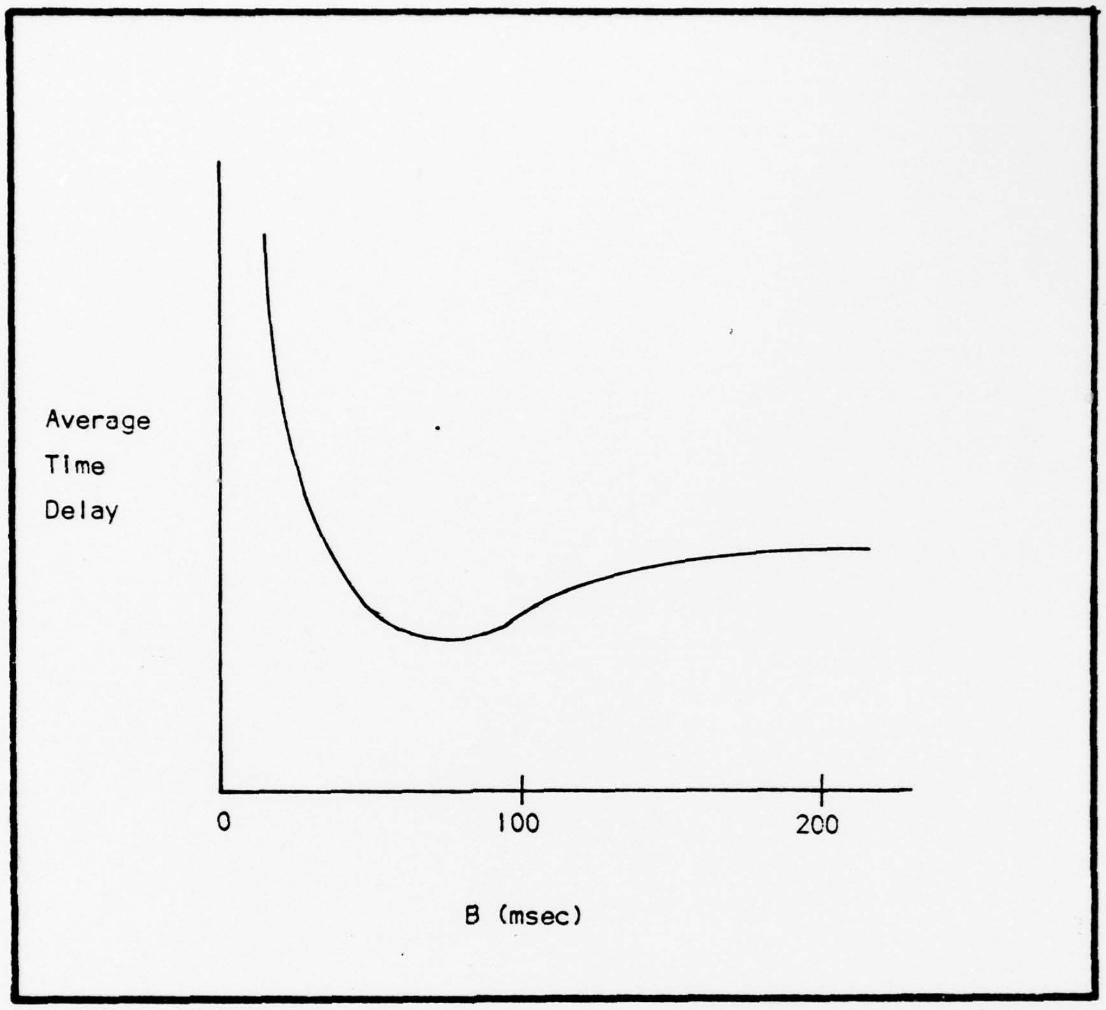
Algorithm Function. The improved reachability algorithm shown in Figure 7 contains the objective function described above and the hold down mechanism described previously. All of the parameters are the same as described in the earlier reachability algorithm (Figure 5) except for the addition of the H matrix and the parameters MIN, MINR, and HOLDT required for hold down. The H matrix is the hold down counter for each destination i, therefore it has i elements, one corresponding to each destination. The MIN parameter stores the temporary value of the best cost to destination i; MINR stores the output line number corresponding to this best cost value. HOLDT contains the initial value of the hold down counters in the H array.

In Step 1 the neighbors cost arrays, C', are read into the NC array in the locations corresponding to the line number on which the, C', array was received. This is identical to the action in step 1 of Figure 5. In Step 2 of Figure 7, the determination is made as to which line corresponds to the best cost to a given destination. The first function of Step 2 is to determine whether hold down

```
Step 1:  Read in neighbors' cost estimates, C', into NC array for
         each destination

Step 2:  For i = 1 to N                      ;Loop through each destina-
                                              tion i.
    2a:     If H(i) > Ø  Then Go To HOLD     ;Goto HOLD if destination i
                                              is in Hold Down.
            Else
            MIN = MAX                        ;Assume destination i is
                                              unreachable.
            MINR = R(i)                      ;Save previous line number
                                              to i.

    2b:     For j = 1 to NL                  ;Loop through line numbers.
    2c:        If NC(i,j)+A(j)+β ≥ MIN Then Go To 2d
                                             ;New delay estimate worse
                                              than old estimate ?
               Else
               MIN = NC(i,j)+A(j)+β          ;Save new estimate.
               MINR = j                      ;Save line number to i.

    2d:        If j ≠ R(i) OR
               NC(i,j)+A(j)+β < C(i) Then Go To NEXTj
                                             ;Checking another line or
                                              new delay better ?
               Else

    2e:        H(i) = HOLDT                  ;Enter Hold Down by setting
                                              hold down timer.

 NEXTj:        Next j

Step 3:     C(i) = MIN                       ;Store best delay after
                                              looking at all lines.
            R(i) = MINR                      ;Store line to destination i
            Go To NEXTi

  HOLD:     C(i) = NC(i,R(i))                ;Continue to update C(i) for
                                              the line to destination i.

    3a:     H(i) = H(i) - 1                  ;Decrement Hold Down counter.

 NEXTi:  Next i

Step 4:  C(self) = Ø                         ;Set cost,line, and hold
         R(self) = Ø                          down timer to zero for
         H(self) = Ø                          node.

Step 5:  Send C array to all adjacent nodes.
```

(Adapted from Algorithm 3-8)(4:240)

Figure 7 Reachability Algorithm With Hold Down



(1:221;21)

Figure 8 Delay Variation with Bias

is in effect for destination i. Since the hold down counter is a decrementing counter, hold down will be in effect as long as H(i) is greater than zero. (Note that if the objective function is *minimizing the path* length, then H(i) is essentially a Boolean variable with HOLDT set to one.) If hold down is in progress for destination i, then the calculation to find the best path (Step 2b) is not performed. Instead, the previous best path to i, R(i), is used to route messages destined for i. Also, if hold down is in progress, the cost to reach destination, i, is still updated using the cost reported on line , R(i), from the adjacent node attached to line, R(i). This is done to purge the old cost information concerning the old best route out of the network.

If hold down is not in progress (H(i) is equal to zero in Step 2a) then the algorithm proceeds almost as before and determines the best path by minimizing the cost to reach destination i along output line j (Step 2c). The improved algorithm solves the timing synchronization problem associated with the previous algorithm by addition of the parameters MIN and MINR. Instead of directly updating the cost array, C, and the readability directory array, R, each time an intermediate better cost and path is found, this algorithm stores the intermediate best path information in MIN and MINR until all of the output lines are investigated for a given destination, i. The C(i) and R(i) values are updated only after the search of all the output lines has been completed (Step 3). This solves the timing problem inherent in the previous reachabilty algoirthm of passing incorrect cost information to adjacent nodes while the output line search is in progress. In addition, the synchronization problem associated with passing to the NOS flow control mechanism incorrect data about the proper output line to use to destination i, is solved because R(i) is not updated until after the new best output is determined for destination i.

In Step 2d the algorithm checks to see if the hold down counter should be initialized for destination i. First a check is made to see if the j loop is checking the node's output line corresponding to the current best path to i. If it is the same then the algorithm checks to see if the condition for initiating hold down is satisfied, which it is if the new delay to destination i on line j is worse than the previous delay, then initiate hold down. Hold down is started in Step 2e by setting the hold down counter for the i^{th} destination to its initial value HOLDT. In Step 3a the hold down counter for the i^{th} destination decremented for each pass through the algorithm. Thus the algorithm can initiate and terminate the hold down state for each destination node in the network separately.

Step 3, as mentioned earlier, assigns the cost to reach destination i and the corresponding output line for this destination to C(i) and R(i) respectively. As noted earlier, this assignment does not occur until after the search of all of the nodes output lines for the i^{th} destination is completed.

Once the C array is finally updated to reflect the status of the node itself (Step 4), the C array can be transmitted to all of the adjacent nodes. An example of the improved reachability algorithm is described in Appendix B.

Decision Process Summary. Both the reachability aspect and the objective function of the decision process have been described in this section. The final result of this description was the generation of the reachability algorithm shown in Figure 7. This reachability algorithm uses a general objective function that minimizes the total delay to a given destination in order to calculate the best path to this given destination. The inputs to the decision process consist of specific parameters that are minimized in the objective function. The outputs of the decision process are the routing directory table (R matrix) containing the output line number associated with each destination and the cost matrix, C, containing the nodes best estimate of the cost to reach any other destination in the network. The specific characteristics of this input and output data and the method for propogating this data through the network is described in the next section.

Updating Process. This section will describe the specific nature of the routing messages passed between nodes in the network. Specifically the message content and message propagation technique will be described.

The overall technique suggested for use in the DEL network is that each node sends routing messages to each adjacent node containing data about its best paths to all destinations. This is information that is destination specific since the cost is given for each destination, but it is source independent since the routing information is not a function of any individual source node in the network. The content and method of propagation of this routing data is described below.

Routing Message Content. In this investigation the contents of the routing messages that are of primary concern are connectivity and delay data. The proposed DEL network reachability algorithm objective function discussed earlier is a general function that minimizes these input data.

Connectivity data is data that defines the physical or topological aspects of the network. In other words, connectivity data is concerned with the number of nodes (or hops) between the source and destination nodes. In the calculation of the shortest path between two nodes, described earlier, the connectivity data was used as the quantitative measurement parameter that was minimized. The basic parameters of the connectivity data are its unit of measurement and the magnitude of this measurement (4:271).

Since the hop count is an integer quantity measuring the number of discrete nodes between any given node and a corresponding destination node, the unit of measurement is one hop. Fractions of hops are not defined.

The magnitude of this hop count can range from one (for adjacent destinations) to N-1 (worst case). The number of bits required to represent this quantity in the routing message is $\log_2(N)$.

In the DEL proposed design the hop count measurement is stored in the cost array, C. This cost represents the number of hops to reach each possible destination in the network. The hop count that is contributed by each node to the total hop count to reach a given destination using any output line is the value one, which is stored in the A(j) array element, where j indicates the output line number. Therefore, if the hop count is used as the content of the routing message it must be contained in the network routing message passed from node to node. Hop count is not the only measurement parameter that can be sent in the routing messages. Another parameter is the total delay to transmit a message from a given node to any destination.

The total delay is a function of three variables (1:215; 4:272):

- Transmission delay - measured in time
- Propagation delay - measured in time
- Node processing delay - measured in time.

The following discussion will describe these delays in terms of unit of measurement, quantitative value, and methods for acquiring the delay values.

The measurement parameter common to all three variables is time. The unit (or smallest) measurement quantity should be the smallest delay expected over any single line. This will enable all of the delay variables to be measured in terms of this smallest unit. In practice, the measurement unit is usually a millisecond. For the DEL laboratory, the suggested measurement unit is also the millisecond since most transmission line characteristics and internal central processor tasks can be measured acturately in milliseconds by most of the minicomputers in the DEL. For example, in the Altair 8800b computer, time increments can be used to measure delay using the Vector Interrupt/RTC hardware module (28).

The quantitative value of the delay variables varies considerably depending on such factors as topology, line speed (bandwidth), and packet size. The transmission delay is the elapsed time it takes to transmit an entire individual packet, therefore transmission delay is affected by packet length and transmission rate. The transmission delay is calculated as follows (4:87):

Transmission delay = (number of bits in packet [bits])/
(transmission rate [bps])

Figure 9 graphically shows how transmission delay varies for given packet sizes. Typical values for the DEL network, using existing hardware and software transmission capabilities are shown in Table 2. Detailed discussions concerning

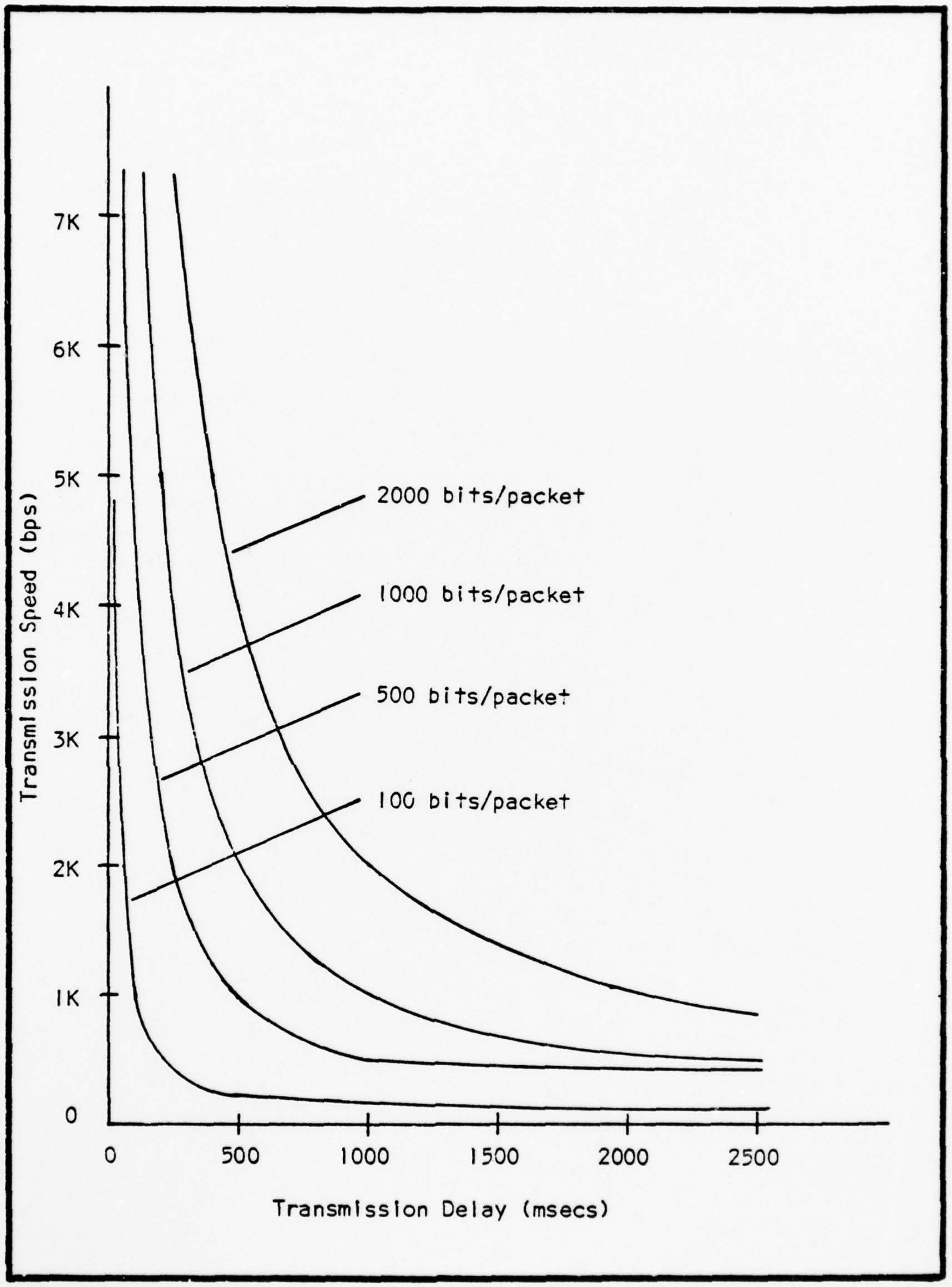


Figure 9 Transmission Delay Characteristics

calculation of line delay and capacity assignments for networks can be found in Schwartz (1), Kleinrock (29; 30), McQuillan (4:86-97), and Metcalfe (31).

Propagation delay is the delay for a given bit of a packet to traverse a path from node to node. Propagation delay then, is clearly a function of the distance traveled. Since electrical signals travel at speeds close to the speed of light, propagation delays are generally insignificant for topologically small networks such as the proposed DEL network. For this reason, propagation delay can be ignored in the DEL design. Typical propagation times are shown in Table 3 (4:89).

Node processing delay is the total time required to process a packet by a node. This time is measured by summing the time required to store (receive) the packet with the time to forward (transmit) the packet plus the time spent waiting in queues because higher priority tasks are being processed by the node (4:87). These times are highly dependent on the type of flow control mechanism used by the node. A description of flow control mechanisms is beyond the scope of this report (Schwartz (1) contains a design approach for constructing flow control mechanisms). However, a typical value for the sum of the three times is one millisecond (4:89) for distributed networks such as the ARPA network. However, depending on the specific design used in implementing the flow control mechanism, this

Table 2

Transmission Speed (bps)	Packed Size (bits)		
	100	500	1200
110	909.1ms	4545.5ms	10.2k ms
300	333.3	1666.7	4.0k
2400	41.7	208.3	500.0
9600	10.4	52.1	125.0
19200	5.2	26.0	62.5

Typical DEL Transmission Delays

Table 3

Distance	Delay	Line Types
100 ft.	.0001ms	Very short (DEL network)
10 mi.	.054ms	intra-city
100 mi.	.54ms	inter-city
3000 mi.	16.2ms	cross-country
45000 mi.	272.0ms	satellite

(Adapted from McQuillan (4:89))

Typical Propagation Delays

delay could vary by several orders of magnitude. Because of this, the nodal processing delay should be included in the DEL routing algorithm.

The total delay then for the DEL network is the sum of the transmission delay and the nodal processing delay. Since the delay values may vary depending on the output line selected for a given destination, each node must calculate the total delay for each output line. This delay is then stored in the node's A matrix (see Figure 7) as the node's delay estimate to send a message out on a given line.

Because of the nature of the reachability algorithm, the delay from node to node, as a message proceeds along any path, is cumulative. The number of bits that must be reserved in the routing message to represent the delay must account for this phenomenon. The number of bits is then:

$$\text{Log}_2[(\text{max path length in hops})*(\text{max delay for the worst node})] + NB$$

where $NB = (\text{max path length in hops}) * \beta$

It is assumed that if the delay is ever greater than MAX then the destination is unreachable. An important point to consider also is the value for MAX. All of the delay considerations above apply to the calculation of the value for MAX. The guideline for assigning the value for MAX is that under normal conditions (no failures) in the network, MAX must be greater than the expected worst case delay to any node in the network.

Now that the delay parameters have been described, the issue still remains of how to obtain the values for the delay. The delay values can, in general, be constant, estimated, or measured at some frequency sufficient to maintaining the accuracy of the data (4:274). If the transmission rate for a given output line of a node is constant then the delay due to transmission can be constant for that line. If the transmission is variable, then some kind of measurement is required. Depending on the sophistication of the communications hardware interface, the line transmission rate could be directly obtainable from the hardware allowing the transmission delay to be readily calculated. Another method for directly measuring the line delay is to periodically measures how long it takes to transmit a fixed length message. An advantage of this technique is that the transmission delay associated with the fixed length packet could be used in evaluating the expected delay for other packets of varying sizes. Since the actual hardware and software design of each node in the DEL network is different, it is difficult to specify which technique to use in measuring transmission delay. The significant point is that some kind of measuring mechanism must be implemented in the proposed design, therefore consideration should be given to this area when designing the overall NOS software. In measuring nodal processing delay, the majority of the delay results from the incoming and outgoing messages being stored in queues waiting for processing. Generally there are separate input and output queues for each of the node's lines, therefore separate measurements can be taken for each line. This measurement is a measure of the expected waiting time for each queue. The delay measurement technique varies depending on the type of queue used (FIFO, priority, circular, etc.). In general, however, the measurement is usually the number of bits to send or the number of packets in the queue. The measurement can be a continual measurement (i.e. instantaneous totals are constantly updated as messages enter and leave the queue) or can be a periodic measurement (i.e. periodic "snapshot" of the queue) taken whenever the routing algorithm needs the data to perform a routing computation (4:275). It is extremely difficult to recommend one technique over the other without the knowledge of the design of the NOS control flow mechanism and queue design. However, from the standpoint of design complexity, the continual measurement technique is generally more complex than the "snapshot" technique. This complexity stems primarily from the fact that the continual method is generally implemented as a parallel process rather than a sequential process. (Coffman (26) discusses the problems associated with parallel versus sequential processing and Schwartz (1) presents a detailed analysis of network queuing and buffer design in his text). Another point to consider in deciding which technique to implement is that continual measurement techniques are generally used in systems where the nodal processing delay varies over a wide range at frequent intervals and it is required to monitor these frequent changes.

This section has described the portion of the updating process concerned with the contents of the routing message. The next section will describe the method used to propagate these contents from node to node.

Routing Message Propagation. This section describes the suggested technique for propagating the routing messages from node to node. As described earlier, the basic ground rule is that propagation of the messages is between adjacent nodes. The propagation method that will be described was assumed to exist in the reachability algorithm of Figure 7 (Steps 1 and 5). Basically the propagation technnique is a method for transferring the node's cost matrix, C, to all adjacent nodes and receiving the adjacent nodes' cost matrices, C'.

The technique suggested for use in the DEL network consists of transmitting the same routing message to each adjacent node at both a fixed periodic update rate and as a result of an asynchronous event trigger. The following paragraphs will describe the justification for this choice and a general description of the design parameters of this technique.

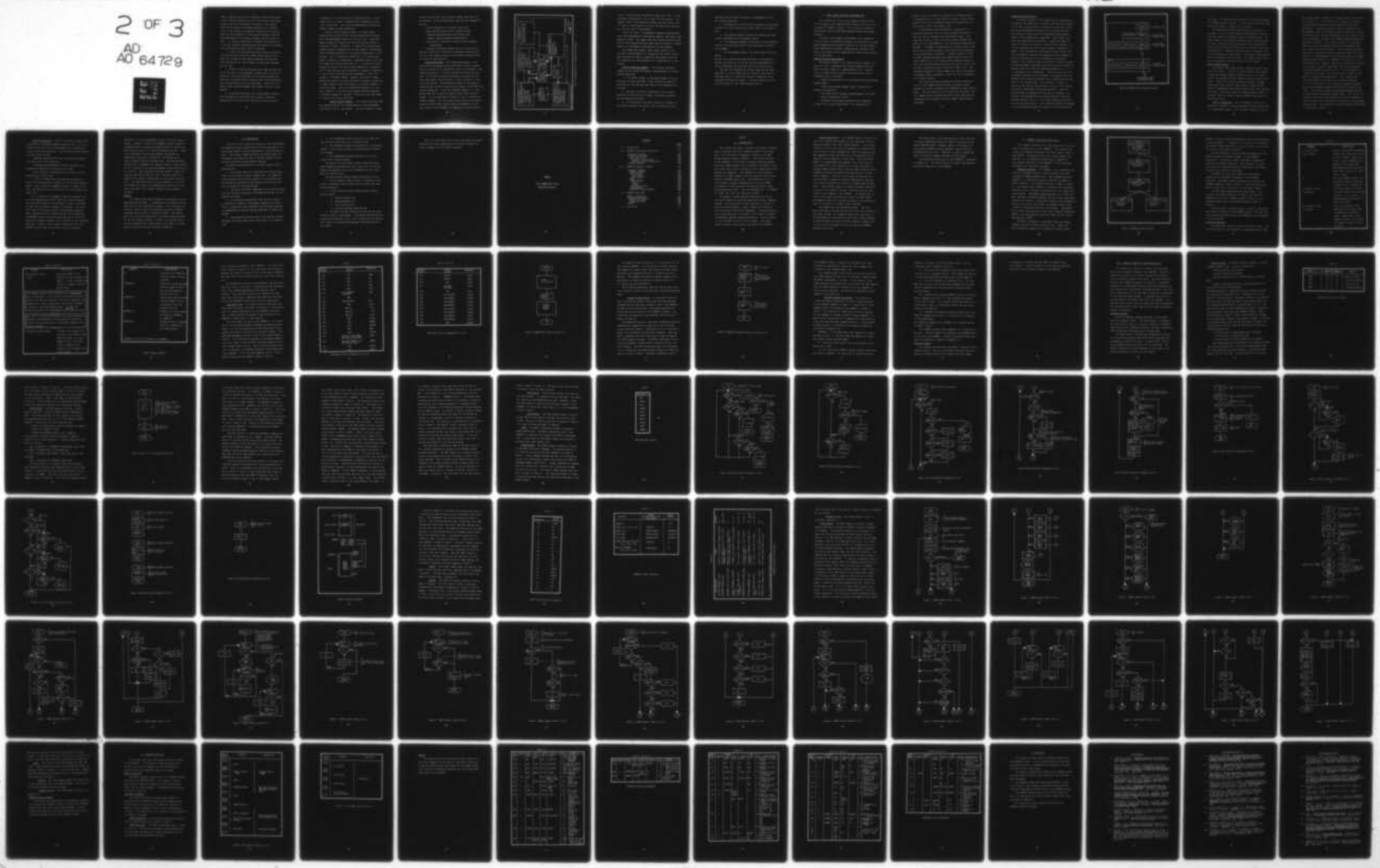
The reason for selecting the same routing message to be sent to each adjacent node was based primarily on the simplicity and minimal memory usage characteristics of this technique. This technique avoids the extra computation required to continually generate different routing messages. Generally the reason for generating different routing messages per output line is to minimize the injection of extraneous information into the network. As already described, the hold down technique added to the basic reachability algorithm also accomplishes this goal. Generating only one routing message also saves considerable memory space since at least one table in memory is required per message per line (4:288). This is especially significant in the DEL network since the majority of the minicomputers in the laboratory have limited memory resources (e.g. generally less than 32k).

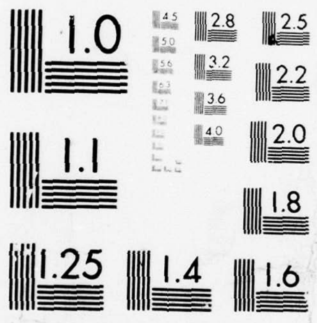
One of the basic requirements of the reachability algorithm is that it be executed frequently enough to detect changes in the network. Periodic execution of the algoirthm by the NOS will provide the means for ensuring that this requirement is satisfied. The actual frequency at which the algorithm is to be executed is a function of many flow variables, therefore it is difficult to suggest a value for the frequency. However, the significant point to consider when selecting the update frequency is that the algorithm must be executed often enough to detect changes in the network. This implies that a priority structure of some kind is required to ensure that the routing algorithm is executed. Specifically, routing information processing must always

AD-A064 729 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO SCH--ETC F/G 17/2
ELECTRICAL ENGINEERING DIGITAL DESIGN LABORATORY COMMUNICATIONS--ETC(U)
DEC 78 D L RAVENSCROFT

UNCLASSIFIED AFIT/GCS/EE/78-16-PT-1 NL

2 OF 3
AD A064729





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

have a higher priority than information packet processing since it may be necessary to alter the current route being used to some new route based on changes in the network. Significant network performance degration will probably occur in the network as congestion develops due to out-of-date routing information if the routing process cannot be executed because of blockage by higher priority tasks (4: 135). In a design application in general and specifically in the DEL routing algorithm design the control flow mechanism of the NOS must ensure that (4:135-136):

1. Input of routing messages must always be possible. This implies that sufficient common buffer space be allocated for storage of the incoming messages and that the input process be run often enough to ensure that messages are not lost.

2. Output of routing messages must always be possible. As in the case of input messages, storage must also be reserved for the output messages and the output message process must be executed often enough so that routing messages can be transmitted as they are generated. (Note that this implies that routing messages have highest priority in the network).

3. As already mentioned, the routing update computation (reachability algorithm) must always be possible. Storage must be available for storing the routing update

information (i.e. the cost and directory matrices, C and R respectively in Figure 7) whenever the reachability algorithm is executing. This can be accomplished by dedicating common storage to these parameters.

Using a periodic update scheme in the NOS control mechanism for activating the routing algorithm will ensure that the requirements above will be satisfied. A periodic update scheme will ensure that the routing calculation is executed at regular intervals. In addition to the periodic update frequency suggested for the DEL network, asynchronous event triggered execution of the routing computation is also desired. This is especially beneficial if a slow periodic update frequency is implemented. Suggested events to be used to trigger the routing algorithm are: (1) send the routing message as soon as the new routing information is computed and (2) send the new routing message if changes are detected in the routing algorithm decision parameters. The first trigger is straight forward. However, the second trigger method may cause the routing algorithm to trigger excessively in large networks since the frequency of network changes is relatively high. Since the proposed DEL network is a very small network, it can safely be assumed that the frequency of network changes will be small relative to the time needed to execute the routing algorithm.

Update Process Summary. This section has described the suggested method for propagating the routing messages from node to node in the network. The flow process is shown

Figure 10 along with the associated common data bases for the process. The following points describe the propagation process:

- Local propagation to adjacent nodes only.
- Same routing message to all adjacent nodes.
- Transmit routing message periodically and
- Transmit routing message asynchronously.
- Use priority structure to ensure routing message transmission
- Allocate sufficient buffer space for routing task.

The next section will describe the message forwarding process(i.e. the general process of forwarding a message from its source to destination).

Forwarding Process. The forwarding process is concerned with the method used by each node to guide a packet from its source to its destination node in the network. The forwarding process is closely related to the decision process. The decision process optimizes and chooses the correct node output line from which to send a packet to a desired destination. The forwarding process is the means of conveying the results of the decision process to the NOS flow control mechanism. In the DEL proposed network it is suggested that this be done by simply using a table look-up scheme. (In the algorithm in Figure 7, this is indeed the scheme assumed). The table should have one entry per destination and each entry should contain the identity of the best output line to forward all packets to a given destina-

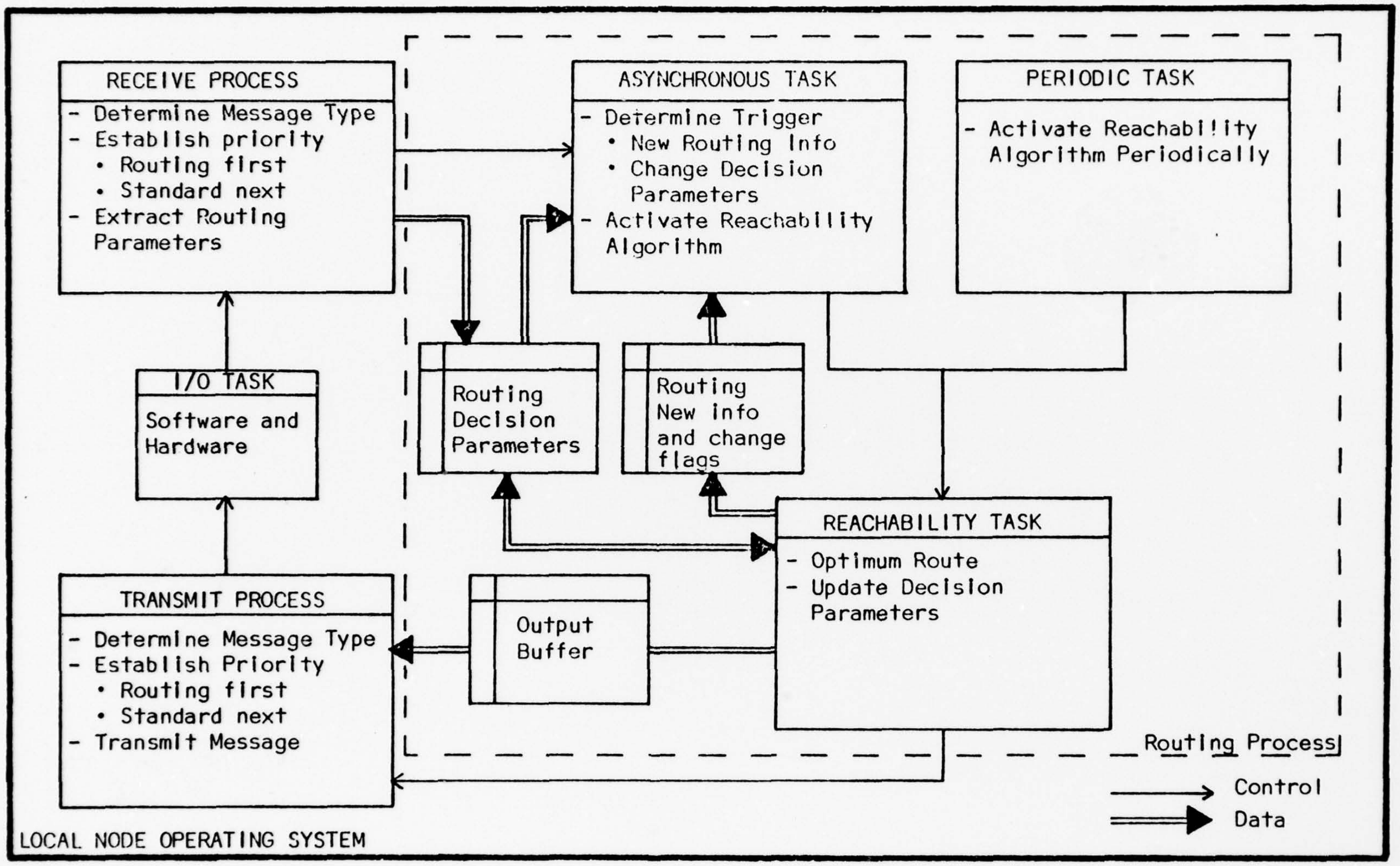


Figure 10 Updating Process Integrated Flow Diagram

tion. In the previous reachability algorithm, (Fig. 7), the directory routing matrix, R, is used for this purpose. The network control flow mechanism simply uses the value in the i^{th} entry in the R matrix to use as the identity of the output line for the i^{th} destination.

Since this table is referenced (indexed) by destination only, the information is destination specific and source independent (4:295). This means that the routing directory table used in the forwarding process need only be N words in length, where N is the number of destinations in the network.

The forwarding process, then, simply consists of a directory routing matrix, R, (containing the output line to each destination) which is passed to the network flow control mechanism to route packets to all destinations in the network.

Routing Algorithm Summary. This section provides a summary of the significant design considerations of the DEL routing algorithm.

1. The control regime is the primary means of classifying the routing algorithm. Since the type of control is distributive, the routing algorithm can be considered distributed.

2. The major functional components of the routing algorithm are the decision process, the updating process, and the forwarding process.

3. In distributed algorithms, adapting to changes in the network topology and traffic flow is relatively slow,

therefore the hold down technique is recommended for the DEL routing algorithm.

4. The objective function of the reachability algorithm minimizes the total delay from a given node to any destination.

5. The updating process consists of sending the same routing information to all adjacent nodes.

6. The proposed propagation technique for the updating process is to use both periodic and asynchronous transmission schemes.

7. The forwarding process is a simple table look procedure.

Overall, the routing algorithm described in the previous section satisfies all of the DEL network requirements and can be integrated into the total NOS with relative ease.

Now that the proposed DEL routing algorithm has been described, a set of communications protocol rules must be developed that will enable the routing information to be propagated through the network. The next chapter will discuss the design of this node-to-node protocol.

IV NODE-TO-NODE PROTOCOL CONSIDERATIONS

This chapter will discuss the suggested specification for the node-to-node communications protocol for use in the proposed DEL computer network. This discussion focuses on the protocol and the associated message format and message contents.

A review of the general requirements for a communications protocol is presented followed by a discussion of a suggested communication protocol hierarchy for the DEL network. After describing the general communications protocol hierarchy, the details of the suggested DEL node-to-node protocol will be described.

General Protocol Requirements

The general pupose of any communications protocol is to provide rules or procedures that enable communication between two ends (nodes of a communications link. Specifically, a communications protocol should provide a set of procedures that (1:323-324):

- enable synchronization to be established and maintained between nodes.
- format the messages between nodes in some kind of standard configuration
- have some kind of message acknowledgement indicators
- detect transmission errors
- terminate the message transmission was completed.

In addition, to these requirements, the communications protocol should be capable of adjusting to different modes of communication (e.g. half-duplex or duplex, polled or non-polled lines) and to different network configurations (e.g. centralized, loops, distributed, etc.).

In implementing these procedures, a protocol will generally establish a set of codes and special characters that are used to implement the requirements above. The codes can consist of many varied and numerous formats depending on the requirements of the associated communications systems. To attempt to standardize the formats used in communications protocols several organizations have developed (or are developing) "standards". Several examples include the ANSI standard Advanced Data Communication Control Procedure (ADCCP) (11), the International Standards Organization (ISO) standard High-Level Data Link Control (HDLC) (32) and IBM's Synchronous Data Link Control (SDLC) (12). The standards attempt to provide common line control procedures and message format standards in order to simplify communication protocol design.

The discussion above described the general characteristics of any communications protocol and gave current examples of several standards for implementing these requirements. The next section of this report will describe a general communication hierarchy and will discuss how these requirements are integrated into the network communications subsystem.

Communications Hierarchy

The primary purpose of the communications subsystem is to provide the capability of transferring a message (packet) (1:14) from a source node to a destination node while satisfying the requirements discussed in the previous section. This process involves passing the message through several layers of the communication subsystem. These layers comprise the communications hierarchy. Figure 10A shows the levels of hierarchy involved in transmitting a message. In the host computer the host-to-host protocol information is added. This information is host-specific and contains control information directing the destination host computer to perform some function on the accompanying transmitted message. This protocol does not necessarily have to conform to the requirements discussed previously since this level of protocol does not effect the message transmission (i.e. it is considered to be transparent). After this the host then calculates and modifies the message in accordance with the protocol that allows the message to be transmitted from the host to its attached node. This protocol is similar in nature to the node-to-node protocol in that it must satisfy the general requirement discussed previously. Once the node received the message from the host, it removes (i.e. extracts the message and host-to-host protocol) the host-to-node protocol since it is no longer needed and attaches the node-to-node protocol. This protocol enables the message to be transmitted through the network to the final destina-

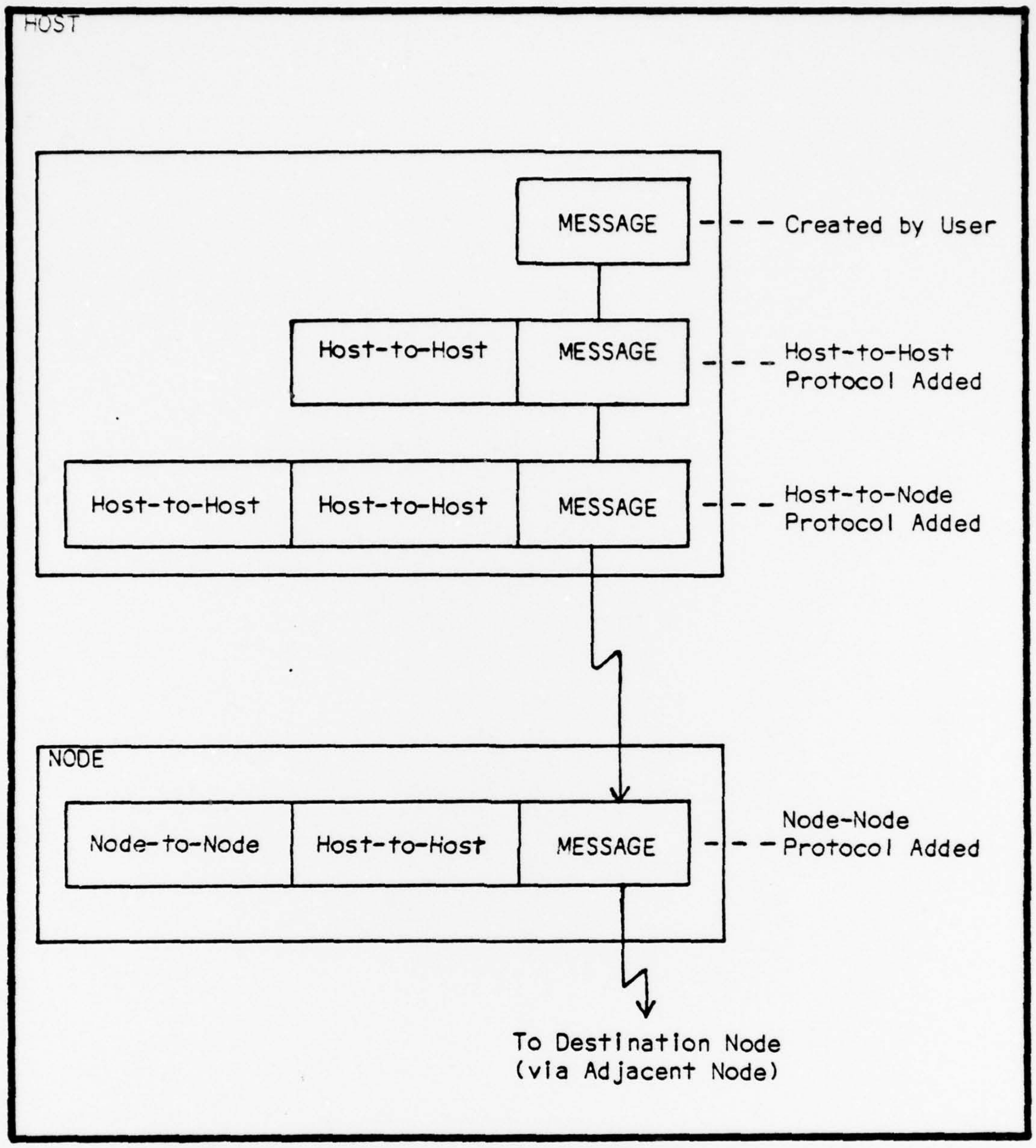


Figure 10A Communications Protocol Hierarchy

tion node. It therefore must satisfy all of the requirements of a communications protocol. At the final destination node, the process shown in Figure 10A is reversed.

The updating and forwarding processes discussed in Chapters II and III use the node-to-node protocol to transmit their routing information along the network paths. The other levels of the protocol hierarchy are used to transfer the user's message from user-to-source host-to-destination host-to-user and are not concerned with the path from source to destination. Therefore, the only level of protocol that will be discussed in this report is the node-to-node protocol which is concerned with this path. This node-to-node protocol is discussed in the next section of this report.

Node-to-Node Protocol

The node-to-node (NTN) protocol contains basically two levels of control. Level-0 is used to control the transfer of messages between adjacent nodes and Level-1 is used to control the message transfer between source and destination nodes. These two levels are necessary because a message being sent from a specific source may be routed through several intermediate nodes before reaching the final destination node. The control required between adjacent nodes is different than the control required between the source and destination nodes.

Level-1 Information. This information controls the sequential transfer of the data message from the source node to its destination node. This includes information such as the message number, request for storage, request for destination status, etc. The information is primarily concerned with end-to-end sequence control between the source and destination. Examples of Level-1 control information are the ARPANET source to destination control mechanism (1:48-49; 3:743-744:32:3-5), the GE Information Services network transmission mechanism (1:19-23), and TYMNET message transmission mechanism (1:30-32). This level of control is transparent (does not effect the message transmission mechanism) data with respect to the NTN protocol. This is because the data contained in the Level-1 protocol does not affect the transmission or acknowledgement of the messages being sent between adjacent nodes since it is intended to be used only by the destination node. However, some of the information contained in the level-1 protocol does affect the routing algorithm discussed previously (See Figure 7). This information includes the node's best cost (the D array in Figure 7) estimate for reaching each destination. This implies that routing messages that contain only routing information (i.e. no host-to-host or user message information present) may be sent between adjacent nodes using only the NTN protocol. This greatly reduces the message overhead required to send routing information between nodes. This Level-1 information (even with the host-to-host and user message present) does not affect the actual transmission of the message between nodes. The information affecting the NTN transmission is contained in Level-0.

Level-0 Information. Level-0 information controls the transmission of messages between adjacent nodes. This information must satisfy the requirement for a transmission protocol discussed previously. For the DEL proposed network, the data should include:

- message synchronization bits to indicate the beginning and end of the transmission.

- a message acknowledgement field to indicate that a message was successfully received by the node.

- error checking information bits to ensure that the message data is error free.

- a field to detect duplicate message transmission.

Several existing protocols satisfy all of the above requirements. These include the ARPANET protocol (1:49-50; 3:744-745), the ADCCP protocol (17:335-339; 11), and the SDLC protocol (12).

It is difficult to recommend a specific protocol withot first designing the overall NOS flow control mechanism. One significant aspect of any protocol is that it conforms to some kind of industry standard if possible. This would help simplify the actual design of the network communications subsystem as well as modifications to the subsystem. The ARPANET protocol mentioned above was one of the first operational network protocols and is similar to the standards mentioned. Because it was created relatively early, the ARPANET protocol does not conform to any one standard, therefore it is not recommended for use in the DEL network design. However, a study of the ARPANET protocol would be extremely useful in aiding the design effort of the DEL network protocol since it is a working network protocol. Additional problems in end-to-end protocol design such as addressing, efficiency, reliability, and timing can be found in Watson (33) and Fletcher (34). The SDLC and ADCCP protocols are "standards" for protocol design. Either protocol will satisfy all of the requirements listed above for the DEL protocol. Both protocols are very similar in their design and selection of one for use in the DEL network will depend on the specific design goals of the overall DEL network. The significant point to be considered is that only one standard be chosen for implementation and that all Protocols designed in the network conform to the selected standard.

Summary

This chapter described the general requirements for any communications protocol. A communication hierarchy was then described for the proposed DEL network. This hierarchy consisted of several levels of protocols required to transmit a message from a source to destination node. The requirements for the node-to-node level of protocol were described for the DEL network protocol. Finally, two protocol standards were suggested for consideration to be used in the DEL network; these are the SDLC and ADCCP protocols.

III CONCLUSIONS

This part of this report has focused on the requirements and suggested design considerations for the development of a distributed network routing algorithm and a node-to-node protocol. This chapter summarizes the significant design considerations that should be used in the DEL network routing algorithm and node-to-node protocol designs.

The design considerations for the DEL network routing algorithm are:

1. The primary means of classifying any routing algorithm is by the type of control regime used. The suggested control mechanism for the DEL network is a distributed control mechanism therefore the DEL network routing algorithm can be considered to be distributed.

2. The major functional components of any routing algorithm are the decision process, the updating process, and the forwarding process.

3. In distributed algorithms, the decision process (i. e. adapting to changes in the network topology and traffic flow) is relatively slow, therefore the hold down technique is recommended for the DEL routing algorithm to improve the process.

4. The objective function used in the decision process minimizes the total delay from a given node to any destination.

5. The recommended updating process is to send the same routing information to all adjacent nodes.

6. The proposed propagation technique for the updating process is to use both periodic and asynchronous transmission schemes.

7. The recommended forwarding process is to use a simple table lookup procedure.

The design of a distributed routing algorithm using these suggested considerations will satisfy all of the DEL network requirements and can be integrated into the total NOS with relative ease.

The node-to-node protocol design considerations are:

1. The DEL network should contain a hierarchy of protocols consisting of host-to-host, host-to-node, and node-to-node protocols.

2. The DEL network proposed node-to-node protocol should contain:

   a. synchronization bits
   b. acknowledgement bits
   c. error checking bits
   d. duplicate message checking bits.

3. The ADCCP and SDLC protocols satisfy the DEL node-to-node- protocol requirements. The node-to-node design considerations presented in Part 1 will provide the DEL network with a means of transmitting messages from node-to-node in the network.

Part 2 of this report will discuss the design and implementation of a data communications interface between the Altair computer and the CYBER 74 computer.

PART 2

Altair 88ØØb/CYBER 74 Data

Interface Description

CONTENTS

		Page
VI.	Introduction	102
VII.	CYBERSCP Functional Description	105
	Software Hierarchy	105
	Functional Overview	107
	High Level Flow	111
	System Initialization	115
	Operator Request Processing	117
	Function Summary	118
VIII.	CYBERSCP Computer Program	120
	Computer Programs	120
	DOSCYB Program	121
	CIN3 Program	123
	CIN3 Inputs	128
	CIN3 Outputs	128
	BASIC	128
	CYBER Program	128
	Inputs	142
	Outputs	142
	Program Listing	146
	TIME Program	146
	CYBERSCP Program Summary	166
IX.	CYBERSCP Data Base	167
	Memory Partioning	167
	Data Base Parameters	167
	DOSCYB Data Base	167
	CYBER Data Base	167
	Summary	168
X.	Conclusion	176

PART 2

VI INTRODUCTION

This chapter describes in general the computer programs used to implement the Altair 8800b to CYBER 74 data interface. The collection of programs is known as the CYBER System Computer Program (CYBERSCP). The primary function of the CYBERSCP is to provide a software interface between the Altair 8800b computer and the ASD/AFIT CDC CYBER 74 computer that allows direct transfer of user created data files between the computers. The CYBERSCP will also allow the execution of SCOPE commands utilizing the facilities of the CYBER 74 INTERCOM System (30). The CYBERSCP consists of five computer programs implemented on the Altair 8800b minicomputer in the Electrical Engineering Digital Engineering Laboratory (DEL). The five programs are: (1) DOSCYB, (2) CYBER, (3) CIN3, (4) TIME and (5) BASIC.

The general system functional operation will be described followed by a detailed description of each computer program. The description of each program will consist of describing each program's inputs, outputs and functional flow (including a functional flowchart). A detailed description of the data base is included in this report following the CYBERSCP computer program description. A user's manual is also provided in Part 3 of this report. This user's manual provides instructions on the use of the CYBERSCP.

General Description. The CYBERSCP System's primary purpose is to support the transfer of user generated files between the CYBER 74 and Altair 8800b computers. This direct transfer of files will allow DEL computers to transfer data files (binary or ASCII) between the CYBER 74 System and the DEL computers without the need for an intermediate form of transfer such as magnetic or paper tape. Since the Altair 8800b is an 8080A based minicomputer, one of the significant benefits of this interface will be the capability to generate 8080A assembly code using the CYBER 74 MAC 80 cross assembler and then to transfer this data directly to the Altair computer. This will allow users the capability to use the CYBER 74 system to generate 8080A machine code and then directly transfer this code to an 8080A based minicomputer. Once the DEL network is created (assuming the Altair is part of this network) the data files may then be transferred to any computer in the network. In essence then, in view of the network description in Part I of this report, the CYBER 74 would be a resource available to the entire network via the Altair to CYBER 74 data interface.

The CYBERSCP has two operational modes. The first mode, DATA mode, provides the user with only local control of the Altair system. All commands used by the user while in the DATA mode are interpreted by the Altair system software. In the DATA mode, the user can write, compile, assemble and run local Altair programs by using the existing CYBERSCP software facilities.

The second mode is the TELE mode which allows the user to directly communicate with the CYBER 74 System using the CYBER INTERCOM control language. While in this mode, all user entered commands, except CONTROL/T, must follow the INTERCOM guidelines (36). CONTROL/T is used to toggle the modes of operation between the DATA and TELE modes.

The functional description of the CYBERSCP is described in Chapter VII. This description will explain the detailed functional operation of the CYBERSCP.

III CYBERSCP Functional Description

This chapter provides a high level overview of the functional characteristics of the CYBERSCP. This overview describes the CYBERSCP functional flow from a user's level of observation. The detailed functions of each computer program (CP) in the CYBERSCP are described in Chapter III. The software hierarchy of the CYBERSCP is first described followed by the functional overview.

Software Hierarchy. The CYBERSCP, is an integrated collection of programs written in both the BASIC high order language and 8080A assembly code. These programs function together in a hiearchical manner to provide the CYBER 74/Altair interface. The software hierachy is shown in Figure 11. The highest level software is defined to be the DOSCYB executive since it controls the operations of all other software programs. The DOSCYB program contains all of the software required to perform diskette input and output operations, diskette file control, input/output control of the peripheral devices such as the line printer and command console (this is done in conjunction with the CIN3 program). The DOSCYB also controls the execution of the BASIC language interperter (BASIC), the input/output controller (CIN3) and the time of day processor (TIME).

The BASIC interpreter is activated using user DOSCYB commands (see User's Manual Section 2.2.2). BASIC uses the input/output capabilities of DOSCYB to transfer data

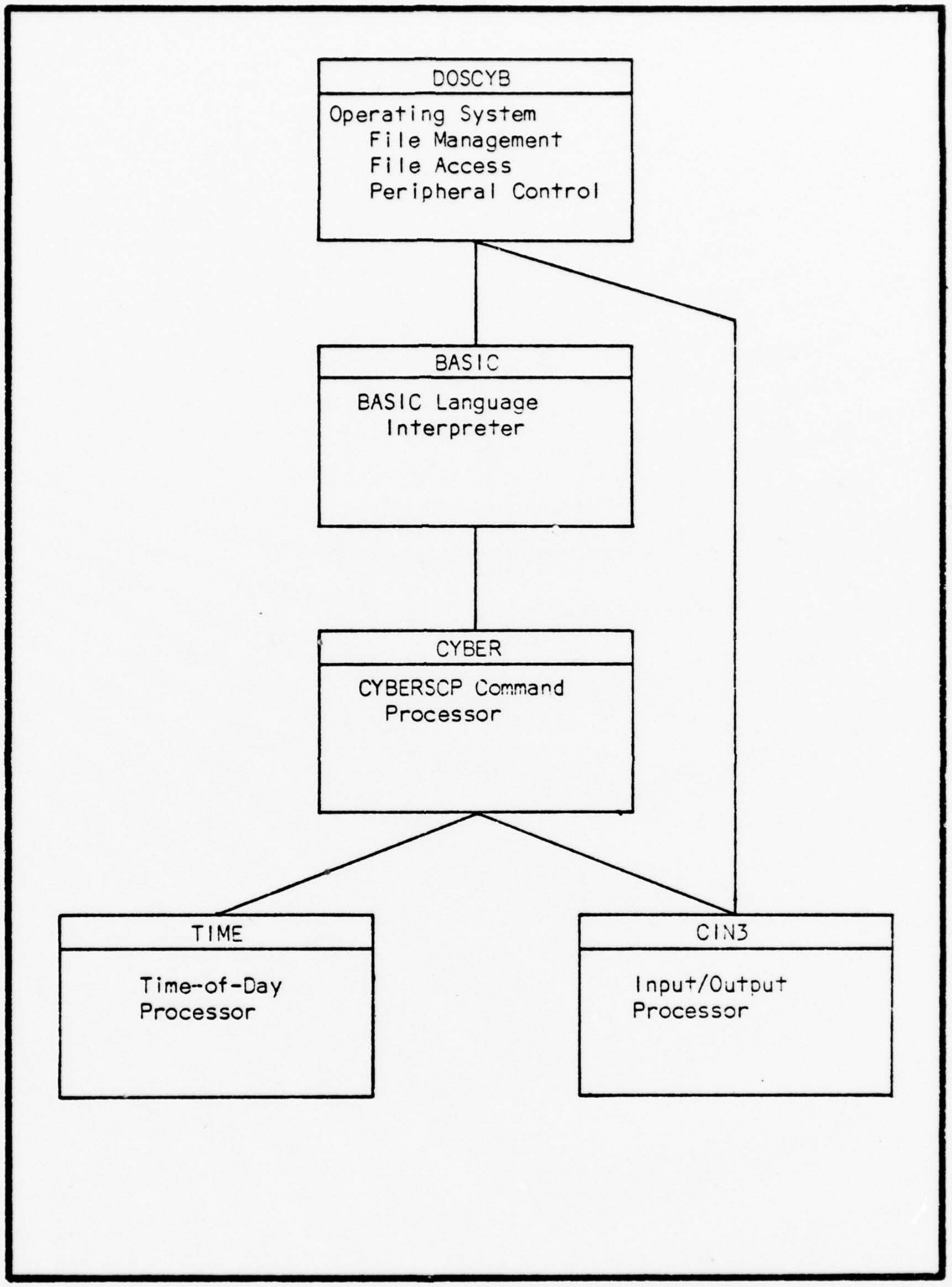


Figure 11 CYBERSCP Software Hierarchy

between the BASIC language programs and the Altair peripherals (including memory). In addition, all diskette file transfers and diskette data manipulation actions are done utilizing the facilities of the DOSCYB executive.

The CIN3 program is the input/output processor for the DOSCYB executive and BASIC program. In the CYBERSCP hierarchy it is directly callable from BASIC language programs and the DOSCYB executive. CIN3 processes all input and output not associated with the diskette for the DOSCYB executive and the BASIC Interpreter.

The CYBER CP is written in BASIC and therefore is subordinate to the BASIC interpreter in the CYBERSCP hierarchy. The CYBER program interprets and executes the CYBERSCP commands listed in Table 4. The use of these commands is explained in Section 3.3 of the User's Manual.

The TIME program determines the time-of-day. It is called by the CYBER program to initiate or set the time parameters using the TI command listed in Table 4. It is therefore subordinate to the CYBER program as shown in Figure 11.

The hierarchical structure shown in Figure 11 represents the levels of software in the CYBERSCP system. The next section describes the operation of the CYBERSCP that corresponds to this hierarchy.

Functional Overview

The functional overview consists of two main areas. The first area that will be discussed is concerned with the high

Table 4

Command	Description
TI[,<hrs>,<mins>,<secs>[,DUR]]	Without the optional parameters returns the system time in hrs:mins:secs format. When <hrs>, <mins>, <secs> parameters are present, sets system time. DUR parameter causes data base item TISET to be set to <hrs> , <mins>, <secs>.
CY	Puts the CYBERSCP system into the monitor state waiting for a CYBER 74 data transmission.
LC,<start>,<stop>[,<device>]	Lists the contents of the input buffer CYFW on the optional <device>. If no <device> is specified then the output is to Port 16 (CRT).
LF,<source>,<stop>[,<device>]	Lists the contents of the diskette file <source> from the beginning to address <stop>, where<stop> is the number of bytes to be listed. The default <device> is Port 16 (CRT).

Table 4 (Cont'd)

Command	Description
SA,<sources>,<dest> [,ADD]	Saves the CYBER 74 file <source> on the diskette file <dest>. If ADD is present the <source> is added to the end of the <dest>.

CAUTION

This command requires that the diskette be <u>unprotected</u> prior to execution. This is done by removing the "write protect" tab from the diskette. Failure to unprotect the diskette will cause a system abort and return to DOSCYB.

```
To return to the CYBER program enter:  JP 2A04 <CR>
                                       RUN <CR>
```

CAUTION

WARNING

When using an unprotected diskette, any failure should be thoroughly investigated and resolved before continuing. Failure to do so could result in permanent damage to the diskette contents.

WARNING

Command	Description
TR,<source>,<dest>[,ADD]	Transfers the diskette file <source> to the CYBER 74 computer and saves it as the CYBER 74 local file <dest>. If ADD is present, the <source> file is added to the end of <dest>.

Table 4 (Cont'd)

Command	Description
BY	Terminates the CYBER program and returns control to BASIC.
CONTROL/Q*	causes data being received from the CYBER 74 (Port 18) to be displayed on the CRT (Port 16).
CONTROL/P*	causes data being received from the CYBER 74 to be displayed on the line printer (Port 34).
CONTROL/S*	Suspends all display data initiated by CONTROL/Q or CONTROL/P.
CONTROL/D*	Terminates the CY command. Control is returned to the CYBER program.

*Command is valid only during a CY command.

CYBER Program Commands

level functional operation of the CYBERSCP. This high level flow is shown in Figure 12. This high level description represents the overall functional logic of the entire CYBERSCP. The lower level flow is derived from this high level description.

The second area that will be described is the low level functional flow. This level is concerned with the functional flow of the CYBERSCP depending on the operator action selected. This low level description, together with the high level description, completely describes the functional flow of the CYBERSCP at the user's level of observation.

High Level Flow. The following section describes the high level functional flow shown in Figure 12. This description is a general description of the CYBERSCP functional flow. A more detailed discussion of the functions of the CYBERSCP is described in a later section.

The first major function of the CYBERSCP is to initailize all of the system computer programs and data base parameters. All of the programs are initialized using the procedures described in Secion 2.2 of the User's Manual. The data base parameters are set to the initial values shown in Table 5. The initialization process is performed automatically as a result of the diskette bootstrap operation (See User's Manual Secion 2.2.1). Once the initialization function is complete, the CYBERSCP System will then accept user input commands. All of the DOS commands (37:5-7) can be entered along with the commands shown in Table 4.

Table 5

Address (Octal)	Content (Octal)	MNUEMUNIC
100	000	MODE
101	015	CR
102	000	EOT
103	012	CRG
104	000	EDIT
105	012,"COMMAND" 055,240	COM
117	000	I
120	012,056,240	EDT
124	000	CYCCR
126	000,001	CYFW
130	000,020	CYCNTI
132	000,020	CYCNT
134	000	CYFULL
135	000	FNAME
146	000	FPRESF
147	000	NMB
154	000	CURLEY
155	012,015,"TELE MODE" 007,000,015,012,000	TMSG
214	012,015,"ABORT CYN", 007,015,012,000	ABMSG
235	001	PRNTTY
236	001	PRNTF

DATA BASE Initial Parameters (1 of 2)

Table 5 (Cont'd)

ADDRESS (Octal)	CONTENT (Octal)	MNUEMUNIC
237	000105	TACOM
241	000120	TAEDT
243	000	TICNT
247	000,000, 005,000	TISET
253	001	TIFLG
254	303,024400	TITRP1
257	000,000000	TITRP2
262	000,000000	TITRP3
265	000,000000	TITRP4
270	000,000000	TITRP5
273	000,000000	TITRP6
276	000,000000	TITRP7
301	000,000000	TITRP8
304	000	INMASK

DATA BASE Initial Parameters (2 of 2)

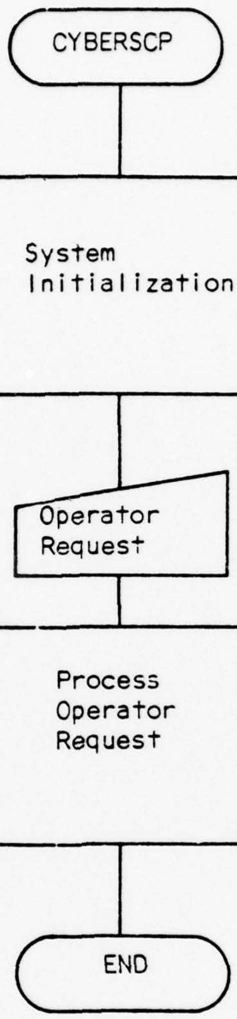


Figure 12 CYBERSCP High Level Functional Flow

The remaining major function is to process each of the user entered commands. This processing includes checking the command for proper syntax and contents and then either performing the requested function or producing an error message. The CYBERSCP continues to process commands until the system is turned off (See User's Manual Section 2.3 for system turn off procedures).

The following subsections describe the low level functions involved in each of the high level functions described above.

System Initialization. As discussed previously, the initialization process initializes both the computer programs and the data base parameters used in the CYBERSCP. The data base initialization is performed automatically during the initialization of the CYBERSCP software. The functional flow describing the software initialization is shown in Figure 13.

The Bootstrap function initializes the DOSCYB executive program which automatically reads the initial data base values from the diskette. (See Section 2.2.1 of the Uers's Manual for the DOSCYB initialization sequence). Next the BASIC interpreter must be initialized in order to execute the CYBER computer program. The BASIC interpreter initialization procedures are described in Section 2.2.2 of the User's Manual. The BASIC interpreter must be initialized prior to executing the CYBER program because the CYBER program is written in BASIC. The BASIC interpreter used in

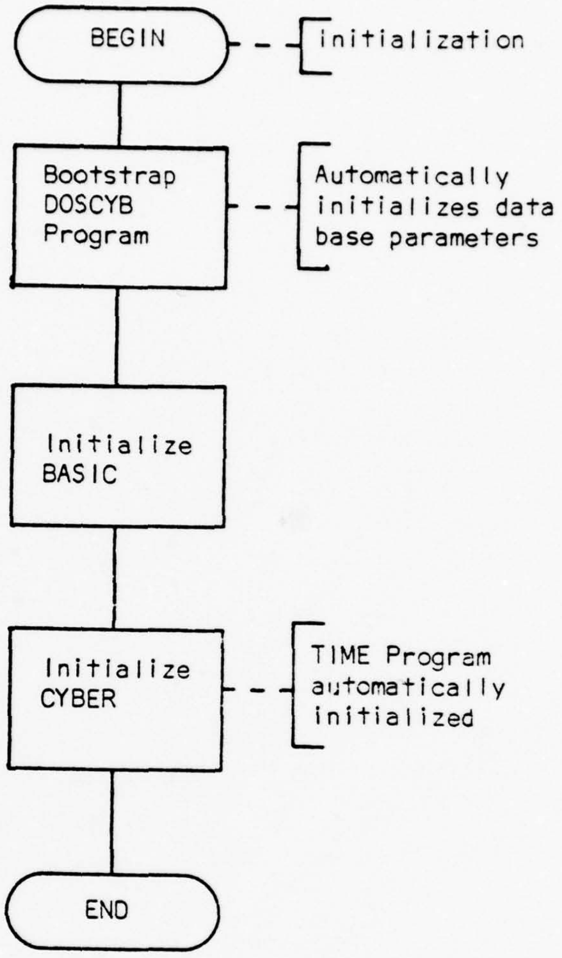

BEGIN
initialization
Bootstrap DOSCYB Program
Automatically initializes data base parameters
Initialize BASIC
Initialize CYBER
TIME Program automatically initialized
END

Figure 13 CYBERSCP Software/data base Initialization Flow

the CYBERSCP system is identical to the North Star Basic interpreter, therefore all North Star BASIC commands are available in the CYBERSCP BASIC (38).

The CYBER program initialization consists of executing the CYBER program (Section 2.2.3 of the User's Manual) after the BASIC interpreter is running. Initialization of the CYBER program also automatically initializes the TIME program. Once the CYBER program is running, all of the commands in Table 4 can be executed. The functional processing of each command is described next.

Operator Request Processing. This section describes the general processing involved in executing each of the CYBERSCP commands in Table 4. A description of the command syntax and purpose of each command is listed in Table 4. This section will describe the actions executed by the CYBERSCP in response to each of these commands. In addition this description will summarize the significant actions resulting from each command execution. A description of these actions is contained in the computer program descriptions in Chapter VIII.

CONTROL/T - In the DATA mode, this command will switch the mode to TELE. In the TELE mode the command will cause the system to enter the DATA mode.

TI - Sets the time of day or sets the duration of the event count, TISET.

CY - Will cause the CYBERSCP system to begin monitoring the CYBER 74 computer. All data that is received from the

CYBER 74 is stored in the CFILE buffer where it can be accessed by other CYBERSCP functions.

LC - Will cause the contents of the input buffer CFILE to print out on a selected device. This command is used to list the data transmitted to the Altair by the CYBER 74.

LF - List the contents of a diskette file. This command can be used to verify that the diskette file contains correct information by listing the file contents onto a selected device.

SA - Transfers the selected CYBER 74 local file to the Altair computer and saves it in the selected destination file on the diskette. The contents of the destination file are written in binary in the exact form as they are transmitted by the CYBER 74.

TR - Transfer the selected diskette source file to the CYBER 74 computer and stores it in the local file selected as the destination.

The remaining commands (BY, CONTROL/Q, P, S and D) are described in Table 4.

The CYBERSCP rocesses each command as it is entered by the user. Once the command is processed, the CYBERSCP waits for the next user command. The CPs that process these commands are described in detail in Chapter III.

Function Summary

This chapter has described the general functions of the CYBERSCP system. Basically the system interprets user commands and transfers data files between the Altair 8800b minicomputer in the DEL and the CYBER 74 computer upon user request. The next chapter will describe the detailed function of each computer program in the CYBERSCP.

VIII CYBERSCP Computer Program Description

This chapter will describe in detail the functional flow of each computer program in the CYBERSCP. Each CP is described in terms of its inputs, outputs and functional flow (including a program flow chart). The description will parallel the CYBERSCP hierarchy in Figure 11. The DOSCYB program will be described along with the CIN3 program since CIN3 contains the input/output interface software. Next the BASIC interpreter will be described followed by a description of the CYBER and TIME programs. A complete description of the data base is included at the end of the chapter. This data base description lists each data base item and describes the characteristics of each item such as use/set parameters, length, and function.

Computer Programs

The first computer program described is the system executive program, DOSCYB. The CIN3 program is included in this description since it is closely related to the DOSCYB. Then following the CYBERSCP hierarchy, the BASIC interpreter, CYBER program and TIME program will be described.

The format used to describe each program will consist of a narrative functional description which describes the accompanying program flow chart. Following this, the computer programs input and outputs will be described. The general description of each parameter is described in the data description at the end of this chapter.

DOSCYB Program. The DOSCYB computer program is written in 8080A assembly code. Its main functions are:

- Perform diskette input/output
- Process diskette file control
- Control execution of all system software including BASIC
- Handle input/output from all system peripherals (in conjunction with the CIN3 program).

The DOSCYB program is a modified version of the North Star Disk Operating System (DOS) (37). Therefore, all of the aspects of DOS discussed in reference 15 apply to the DOSCYB. The modification to DOS were made following the guidelines for personalizing DOS found in the DOS manual (37:9). The modifications made to DOS accomodate the specific input/output conventions required by the CYBERSCP System. These modifications are shown in Table 6 and provide the linkage to the input/output routine, CIN3 stored in PROM. In addition to the modification that provide input/output linkage, the initialization routine in DOS was modified to provide the following capabilities.

- initialization of input/output ports 024_8-027_8.
- initialization of output port 042.
- initialization of the file DATA on the diskette.

The input/output port modifications are straight forward and follow the procedures in the 88-2SIO Board Manual (39) for ports 024_8-027_8 and the procedures in the 88-PIO Board Manual (40) for port 042. The modification to initialize

Table 6

Memory Location (HEX)	Contents (HEX)		Comments
	DOS	DOSCYB	
200E	58	67	Output routine
200F	7E	FD	starting address
2011	00	60	Input routine
2012	29	7E	starting address

DOS Modifications for DOSCYB

the file DATA is shown in Figure 14. After the 8080 registers are initialized, the DOS routine DCOM is called to transfer the data from the diskette to RAM starting at the location in the DE register. DCOM is described in the DOS manual (32:14-16). Once the DATA file is transferred, DOSCYB jumps to the CIN3 routine to monitor the input/output devices.

CIN3 Program. The CIN3 routine is written in 8080A machine code and is stored in PROM locations FB00(HEX) to FDFF. The primary function of this routine is to process the input/output functions for all of the CYBERSCP System programs. Specifically the CIN3 functions are:

- handle all input/output (except diskette) for DOSCYB
- process all CYBER 74 inputs
- process the CONTROL/P,Q,S,D, and T commands.

The processing of these specific functions is shown in Figure 15 and is described below. CIN3 has three main functions as follows:

CIN - processes all CYBERSCP inputs except input from the CYBER 74 computer when in the DATA mode.

CYIN - processes the CYBER 74 input data when in the DATA mode.

COUT - processes all CYBERSCP output data.

The CIN function is a software input device handler that sequentially queries each enabled device in the system. A device is enabled by selecting the proper device code and storing it in INMASK. The logic flow for each value of INMASK is shown in Figure 15. Port 020 (the command console)

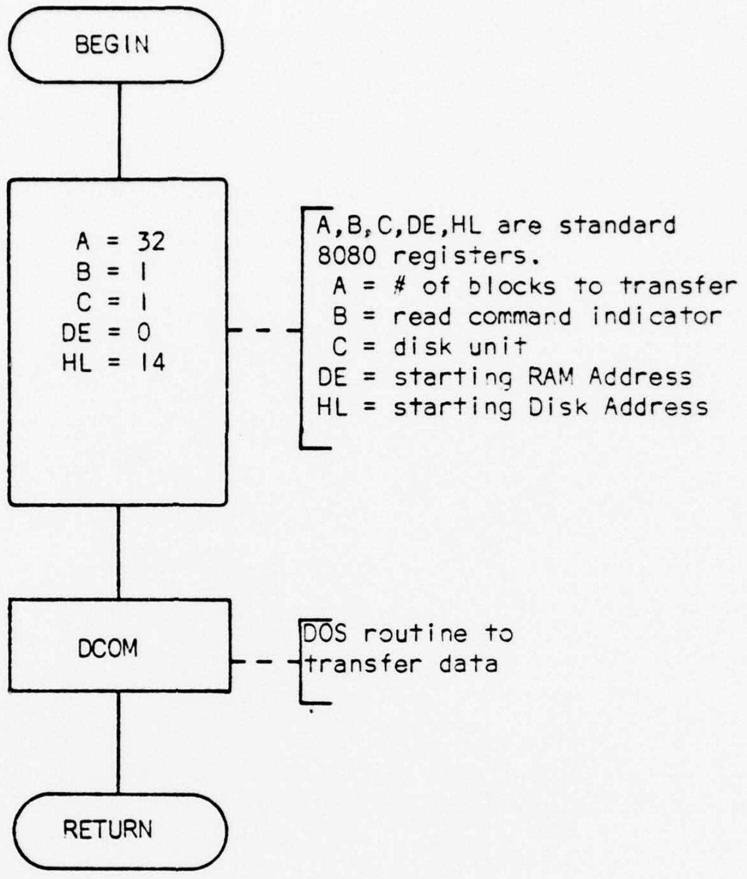

BEGIN
A = 32
B = 1
C = 1
DE = 0
HL = 14
A,B,C,DE,HL are standard 8080 registers.
A = # of blocks to transfer
B = read command indicator
C = disk unit
DE = starting RAM Address
HL = starting Disk Address
DCOM
DOS routine to transfer data
RETURN

Figure 14 DATA File Initialization Modifications

can always input data into the system regardless of what device has been selected. For example, if INMASK is equal to 1 then the device connected to Port Q24 may input data. However, as shown in the flowchart, Port Q2Q may also input data since it is always enabled. The CONTROL/T command is also processed by CIN. CONTROL/T can be input only by the command console (Port 020) as shown in Figure 15. Since CIN can only be called from the DOSCYB executive while in the DATA mode, a CONTROL/T will force the CYBERSCP into the TELE mode. While in the TELE mode, the processing of the CYBER 74 input is done in CIN. Control is not returned to DOSCYB until another CONTROL/T is entered putting the CYBERSCP back into the DATA mode.

The CYIN function is called by the BASIC language program CYBER in response to a CY command. The CYIN routine, upon exit, returns control back to the CYBER program. The primary purpose of the CYIN routine is to process the data sent by the CYBER 74 computer. This processing includes automatic end-of-message detection, storing of the received data into the CFILE input buffer, and processing of the CONTROL/P,Q,S and D commands.

The first action of the CYIN function is to enable the interrupt system in the event that it was previously disabled. Since it is desirable to allow the user of the system to remain in control of the system even while data is being transferred from the CYBER 74 computer, the CYIN routine constantly checks to see if the command console

(Port Q20) has an input ready. This allows the operator to control the CYBER 74 data transfer process by allowing input of the CONTROL/D,P,Q and S commands. The function of each command is described in Table 4. The functional flow of the command processing is shown in Figure 15. The CYBER 74 input data is input into the CFILE input buffer. This buffer is a fixed length buffer containing CYCNT number of words. The buffer is shown in Figure 16. The address of the first word of the buffer is stored in CYFW. CYCUR is the relative address of the last word entered into the buffer. The effective address is the sum of the base address and the relative address (CYFW + CYCUR). The maximum number of words allowed to be stored in CFILE is CYNTI. CYCNT is initially set to CYCNTI and is decremented each time a word is added to the buffer. Attempting to store more than CYCNT number of words (CYNCT < 0) in CFILE will set the buffer overflow flag, CYFULL, to one. Once the CY command is entered by the operator all data transmitted by the CYBER 74 (and before the buffer is full) will be stored into the buffer. The received data is stored sequentially, starting at location CYFW, until an end-of-text delimiter is encountered, a CONTROL/D is entered, or the buffer is filled. The end-of-text delimiter can consist of two different character strings depending on the mode of the CYBER 74 computer. If the CYBER 74 computer is in the EDITOR mode (36) then the delimiter consists of a line feed character, a period, and a "blank" (i.e. 012_8, 056_8, 240_8). The special "blank" character (240) is not transmitted by the CYBER. It is a special character that signifies the end of the delimiter string stored in the DOSCYB data base in the variable EDT. If the CYBER 74 is not in the EDITOR mode, then the end-of-text string is: "COMMAND-(blank)". The method used to detect the end-of-text string is illustrated in Figure 15. Specifically, each incoming character is compared against the first character of the desired delimiter string depending on the CYBER 74 mode. If a match is found, a pointer (TACOM for the not EDITOR delimiter and TAEDT for the EDITOR delimiter) is incremented to point at the next character in the string, EDT. This process is continued until a character fails to match or the special "blank" character (240_8) is found. If a character fails to match and it was not the "blank" then the pointers are reset to the beginnning of the delimiter string and the search starts again. If the delimiter is found, then the end-of-text found flag (EOT) is set and CYIN returns to the calling program.

The third major function of CIN3 is the COUT routine. This routine processes all output of the CYBERSCP except diskette transfers. The COUT routine is a software device handler that sequentially tests the status of each of the output devices. COUT is called by the DOSCYB executive in response to any output request (except diskette output requests) from the CYBERSCP system. The device code used to select an output device is passed to the COUT routine in the 8080A A register. The functional flow for each device code is shown in Figure 15. The device codes are described in Section 3.0 of the User's Manual.

CIN3 Inputs. There are two classes of inputs for the CIN3 program: (1) operator and (2) data base. The operator inputs are the CONTROL/P,Q,S,D and T commands. Since the CIN3 routine is part of the low level software in the CYBERSCP, it has as data base inputs, all of the parameters listed in Table 7.

CIN3 Outputs. The CIN3 program output is the received CYBER 74 data which is written into the file CFILE in RAM. The data in CFILE is written in sequential order as it is received from the CYBER 74 computer.

BASIC. The BASIC interpreter program is written in 8080A machine code. The BASIC interpreter used in the CYBERSCP is identical to the North Star BASIC interpreter (38). A description of the BASIC program can be found in the North Star Basic Manual (38).

CYBER Program. The CYBER computer program is written in the BASIC high order language. Its primary function is to interpret and execute the user commands indicated in Table 4. These commands provide the user with file manipulation, file maintenance, system time control and CYBER 74 INTERCOM command capabilities. Illegal commands and commands that are syntactically incorrect are rejected and flagged as errors to the user. The overall CYBER program functional flow is shown in Figure 17. The succeeding sheets of Figure 17 show the detailed flow of the subroutines that make up the CYBER program.

Table 7

MNUEMONIC
EDT
CYCUR
CYFW
CYCNTI
CYCNT
PRNTTY
PRNTF
TACOM
TAEDT
INMASK

CIN3 Data Base Inputs

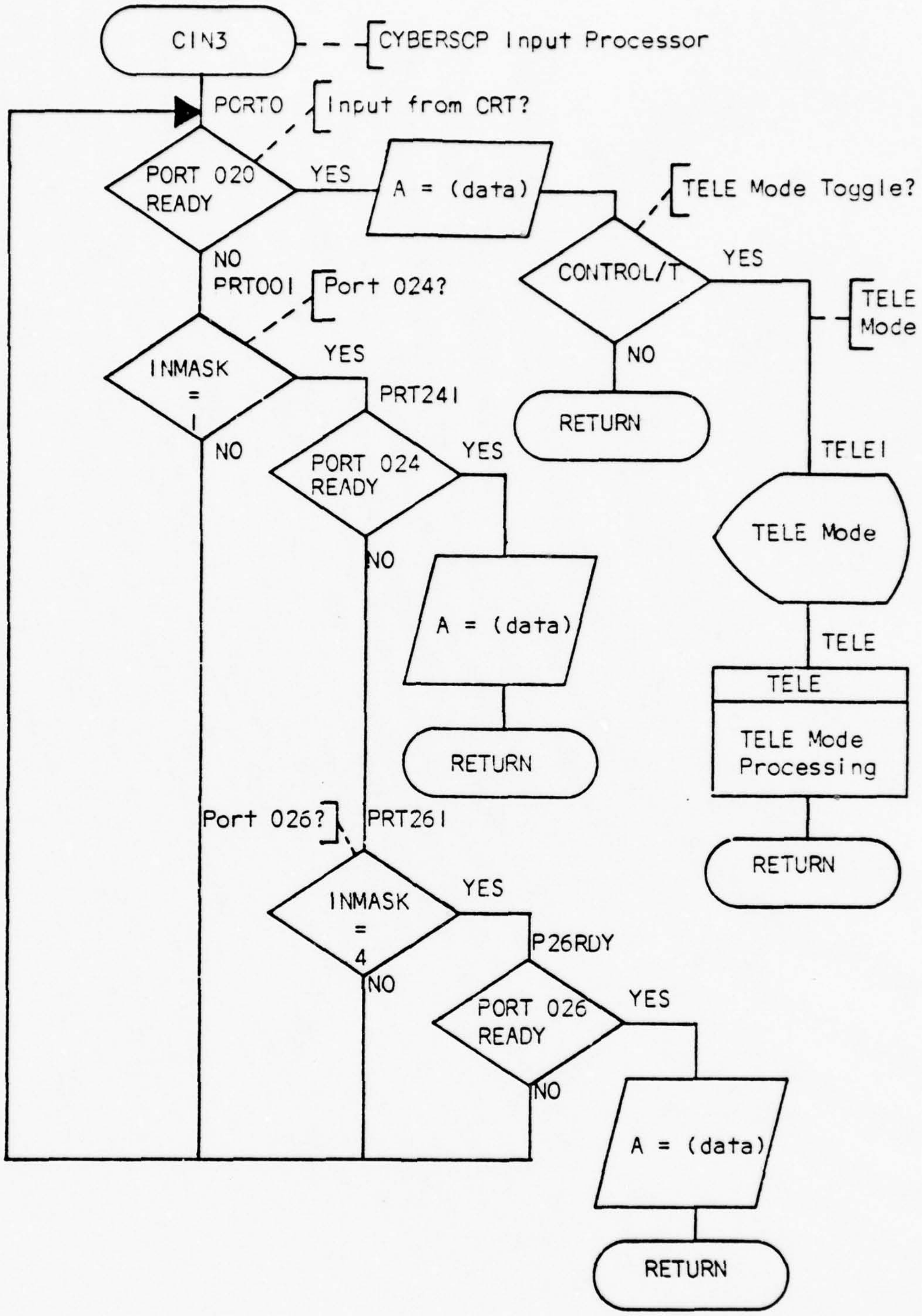


Figure I5 CIN3 Input/Output Processing (I of I0)

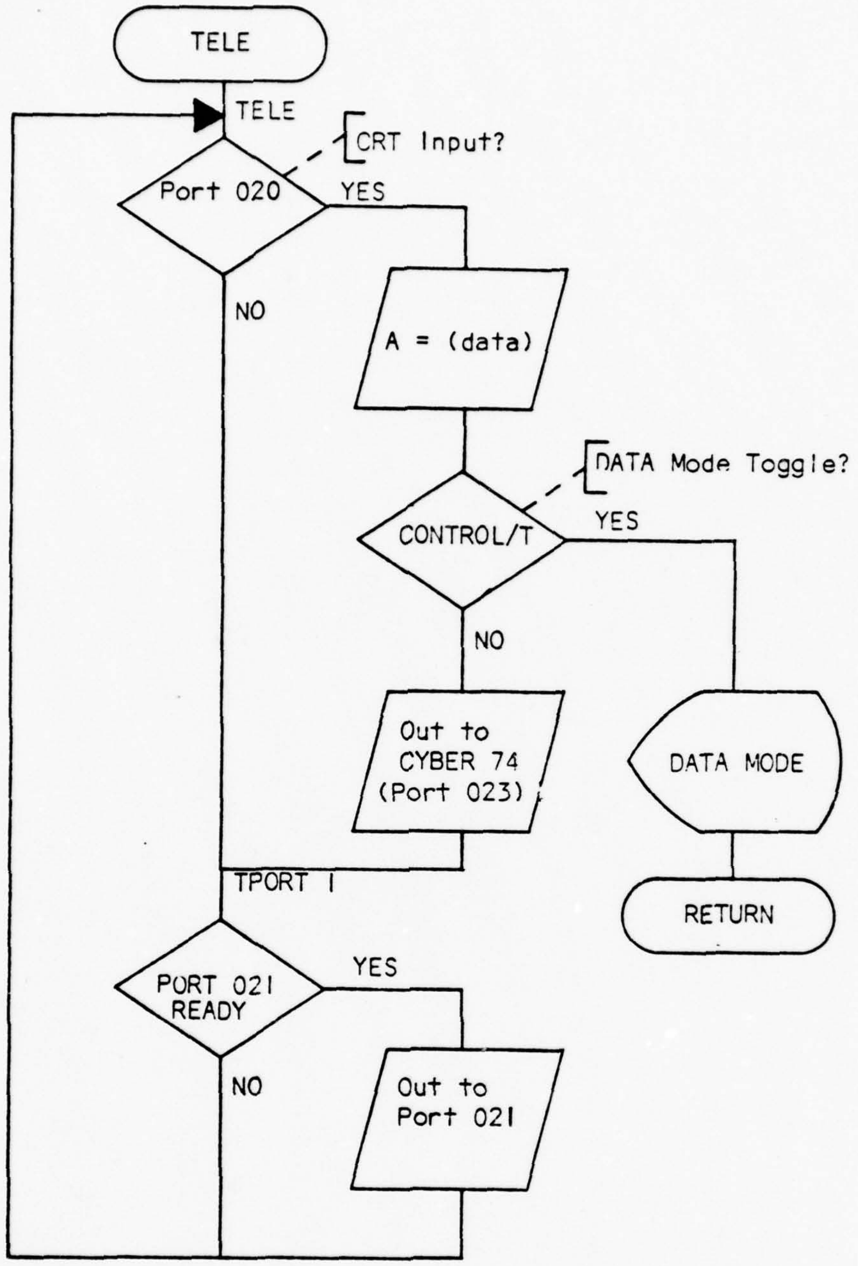


Figure 15 CIN3 Input/Output Processing (2 of 10)

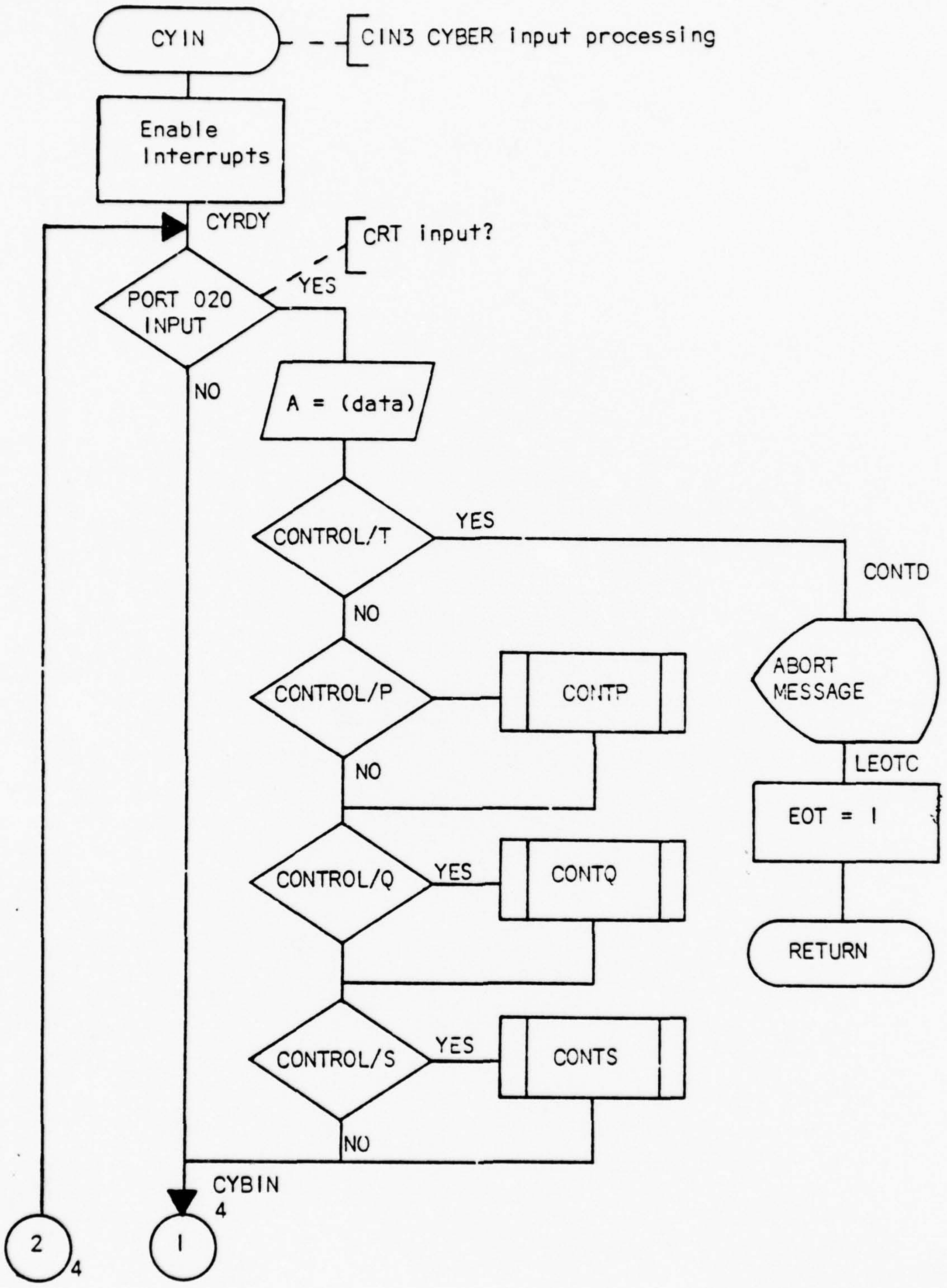


Figure 15 CIN3 Input/Output Processing (3 of 10)

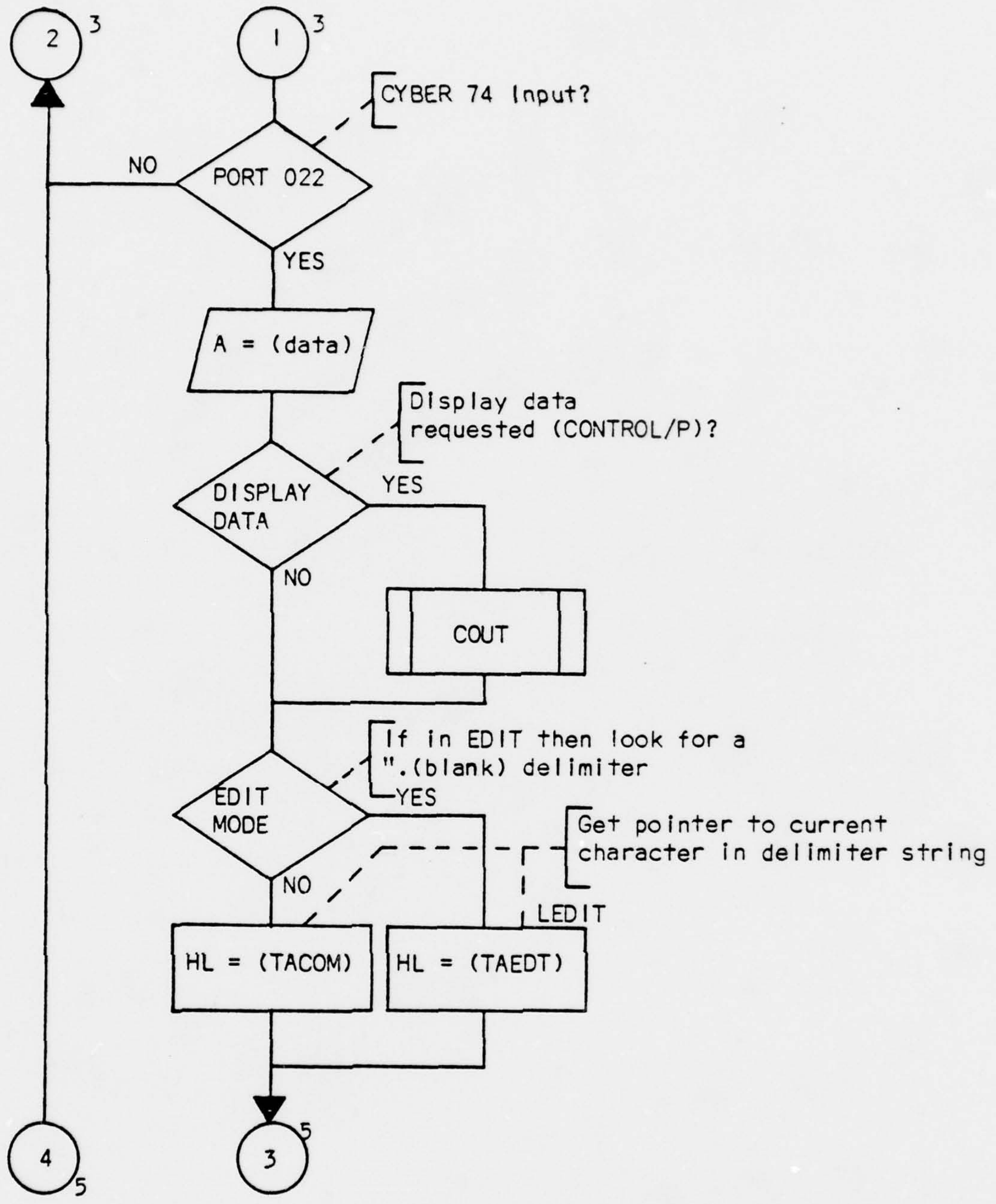
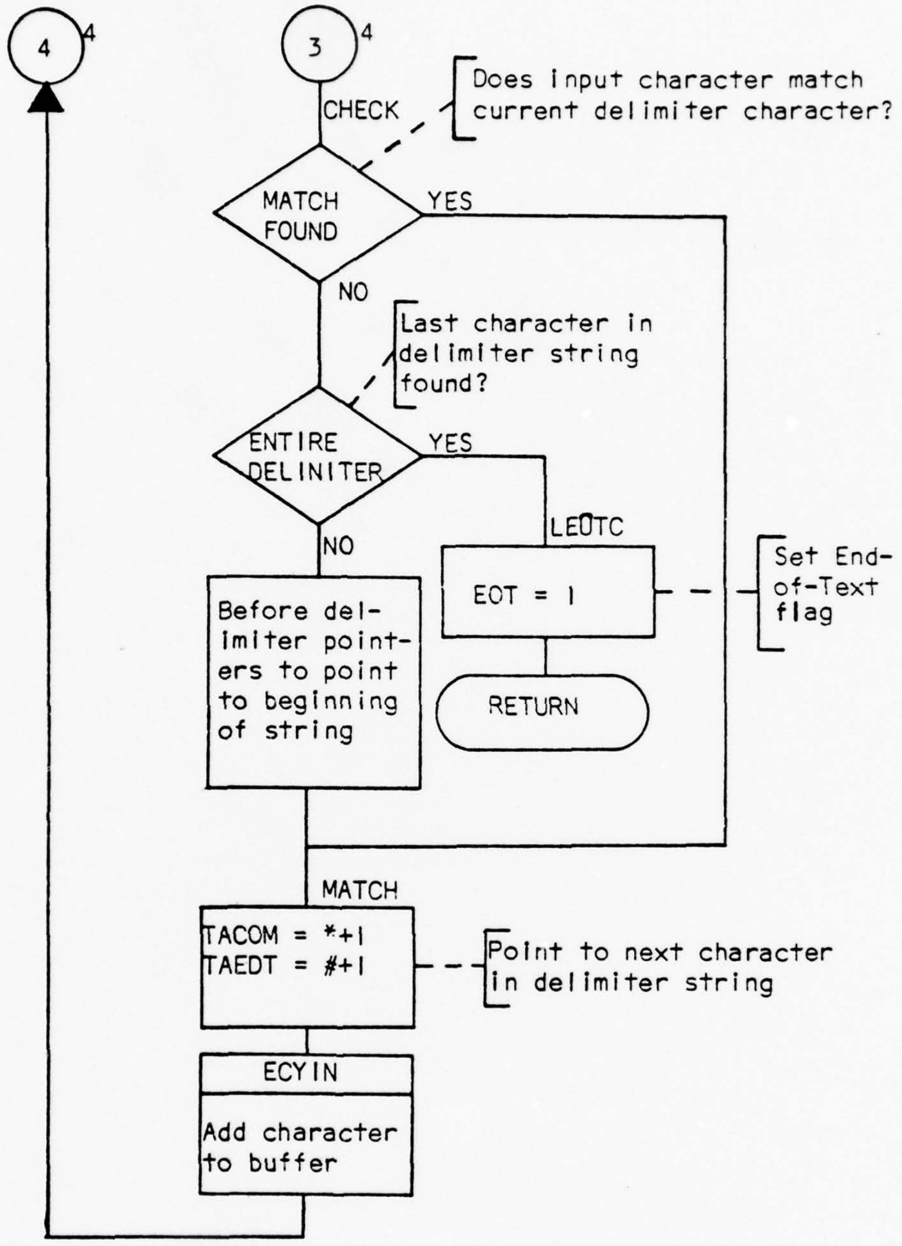

2
3
1
3
CYBER 74 Input?
NO
PORT 022
YES
A = (data)
Display data requested (CONTROL/P)?
DISPLAY DATA
YES
NO
COUT
If in EDIT then look for a ".(blank) delimiter
YES
EDIT MODE
Get pointer to current character in delimiter string
NO
LEDIT
HL = (TACOM)
HL = (TAEDT)
4
5
3
5

Figure 15 CIN3 Input/Output Processing (4 of 10)



4
4
3
4
CHECK
Does input character match current delimiter character?
MATCH FOUND
YES
NO
Last character in delimiter string found?
ENTIRE DELINITER
YES
LEOTC
NO
EOT = 1
Set End-of-Text flag
Before delimiter pointers to point to beginning of string
RETURN
MATCH
TACOM = *+1
TAEDT = #+1
Point to next character in delimiter string
ECYIN
Add character to buffer

Figure 15 CIN3 Input/Output Processing (5 of 10)

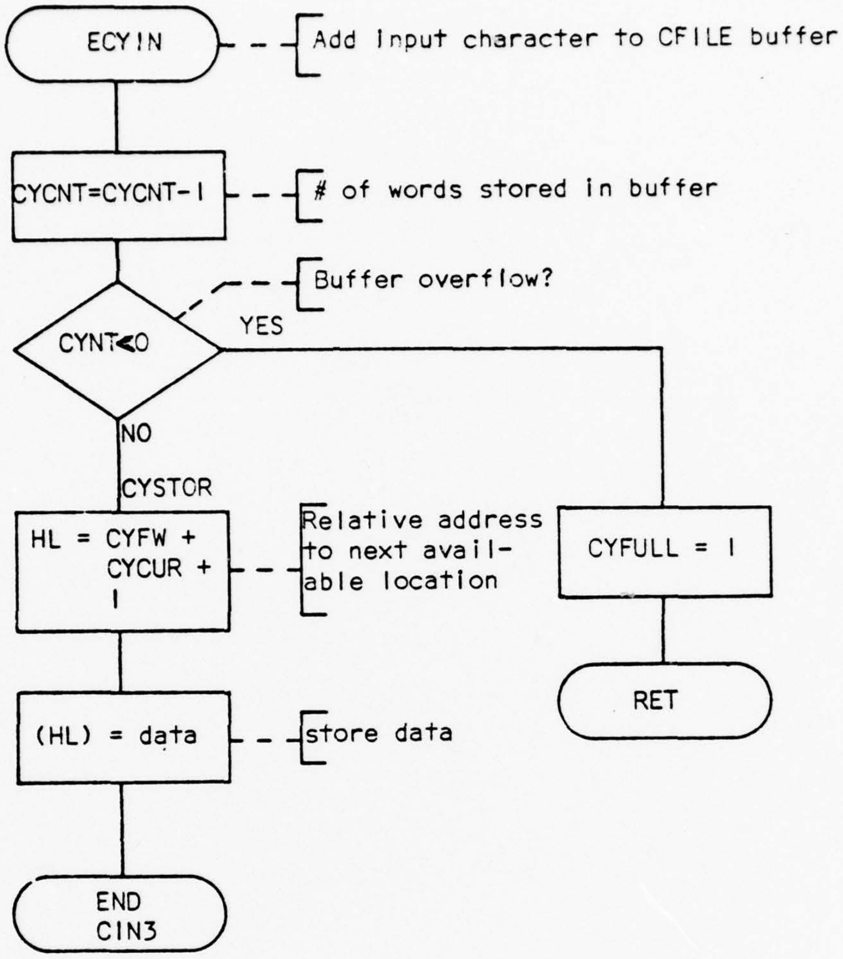


Figure 15 CIN3 Input/Output Processing (6 of 10)

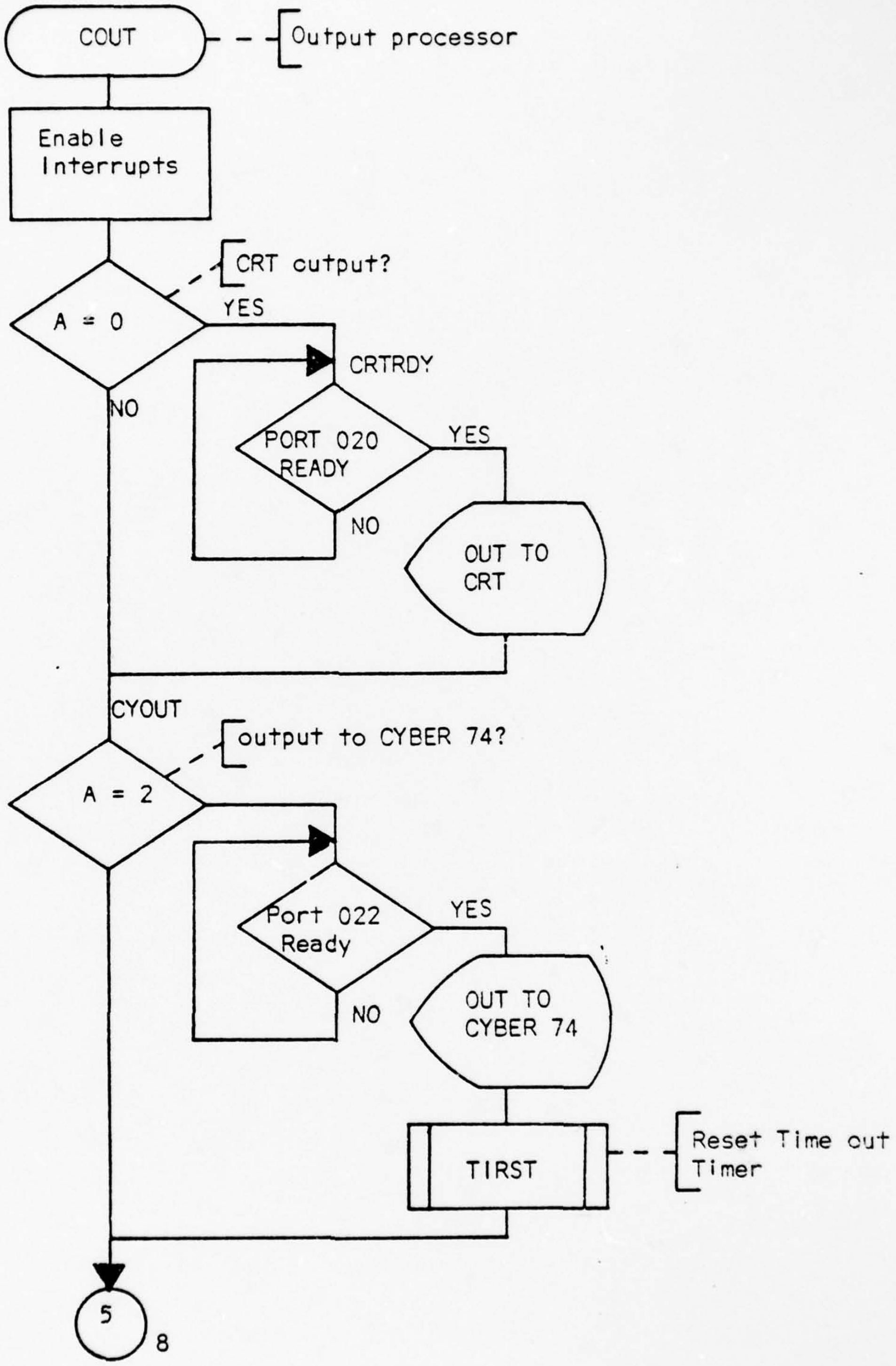


Figure 15 CIN3 Input/Output Processing (7 of 10)

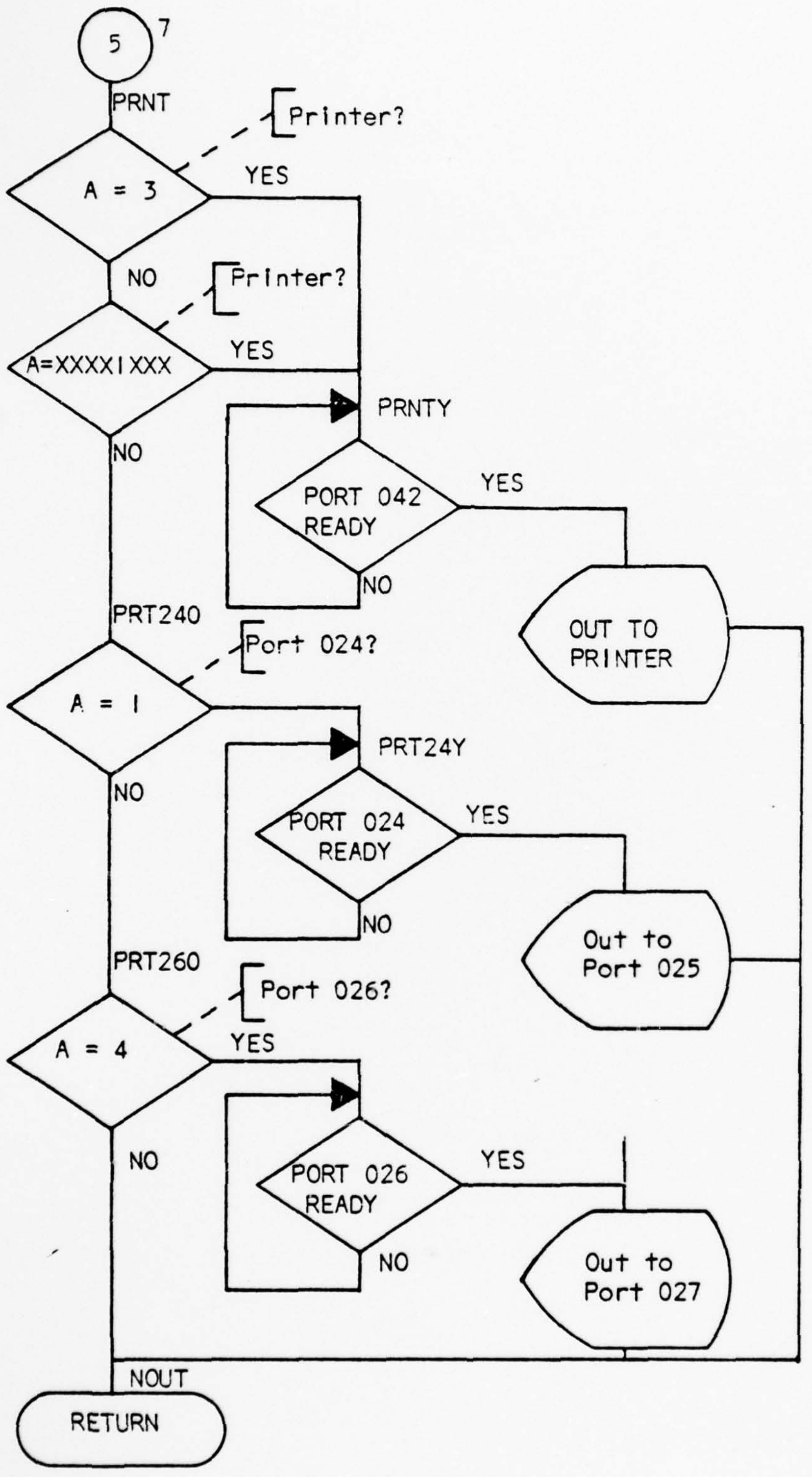


Figure 15 CIN3 Input/Output Processing (8 of 10)

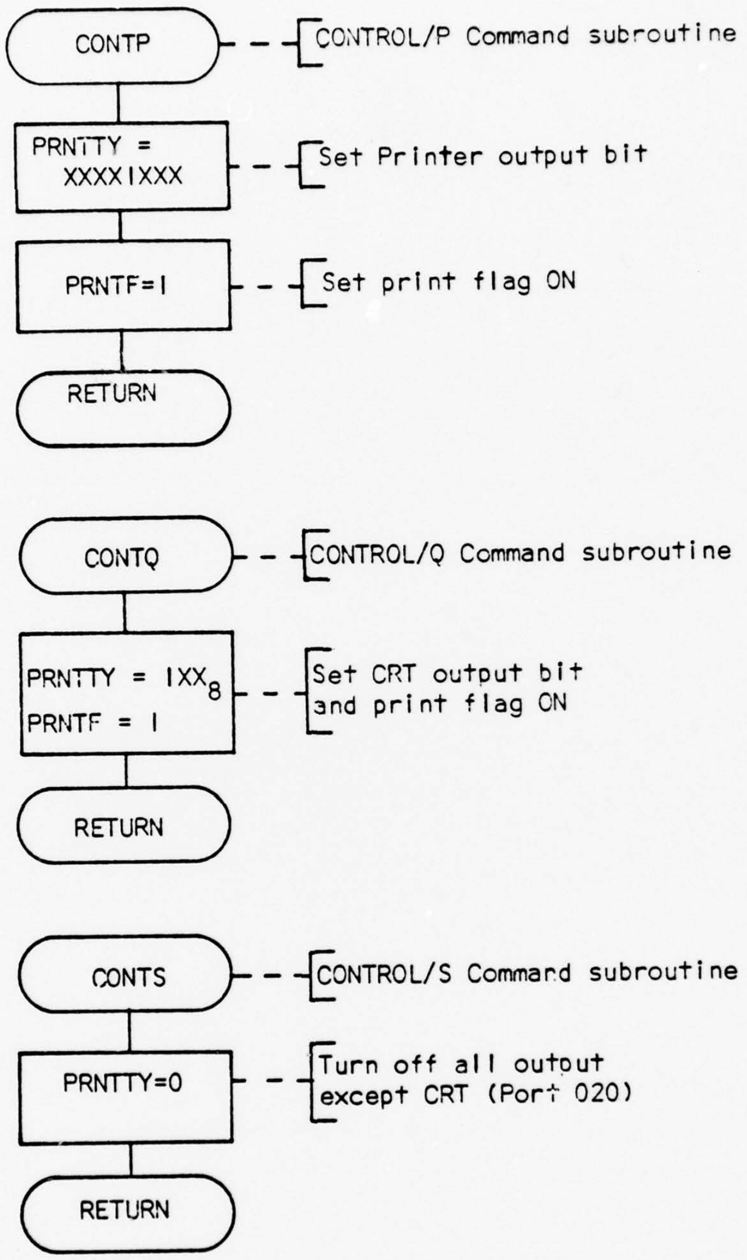

CONTP
CONTROL/P Command subroutine
PRNTTY = XXXX1XXX
Set Printer output bit
PRNTF=1
Set print flag ON
RETURN
CONTQ
CONTROL/Q Command subroutine
PRNTTY = 1XX8
PRNTF = 1
Set CRT output bit and print flag ON
RETURN
CONTS
CONTROL/S Command subroutine
PRNTTY=0
Turn off all output except CRT (Port 020)
RETURN

Figure 15 CIN3 Input/Output Processing (9 of 10)

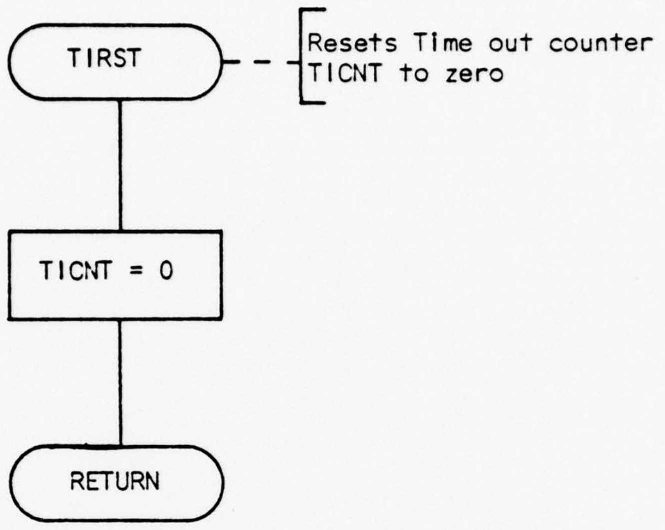


Figure 15 CIN3 Input/Output Processing (10 of 10)

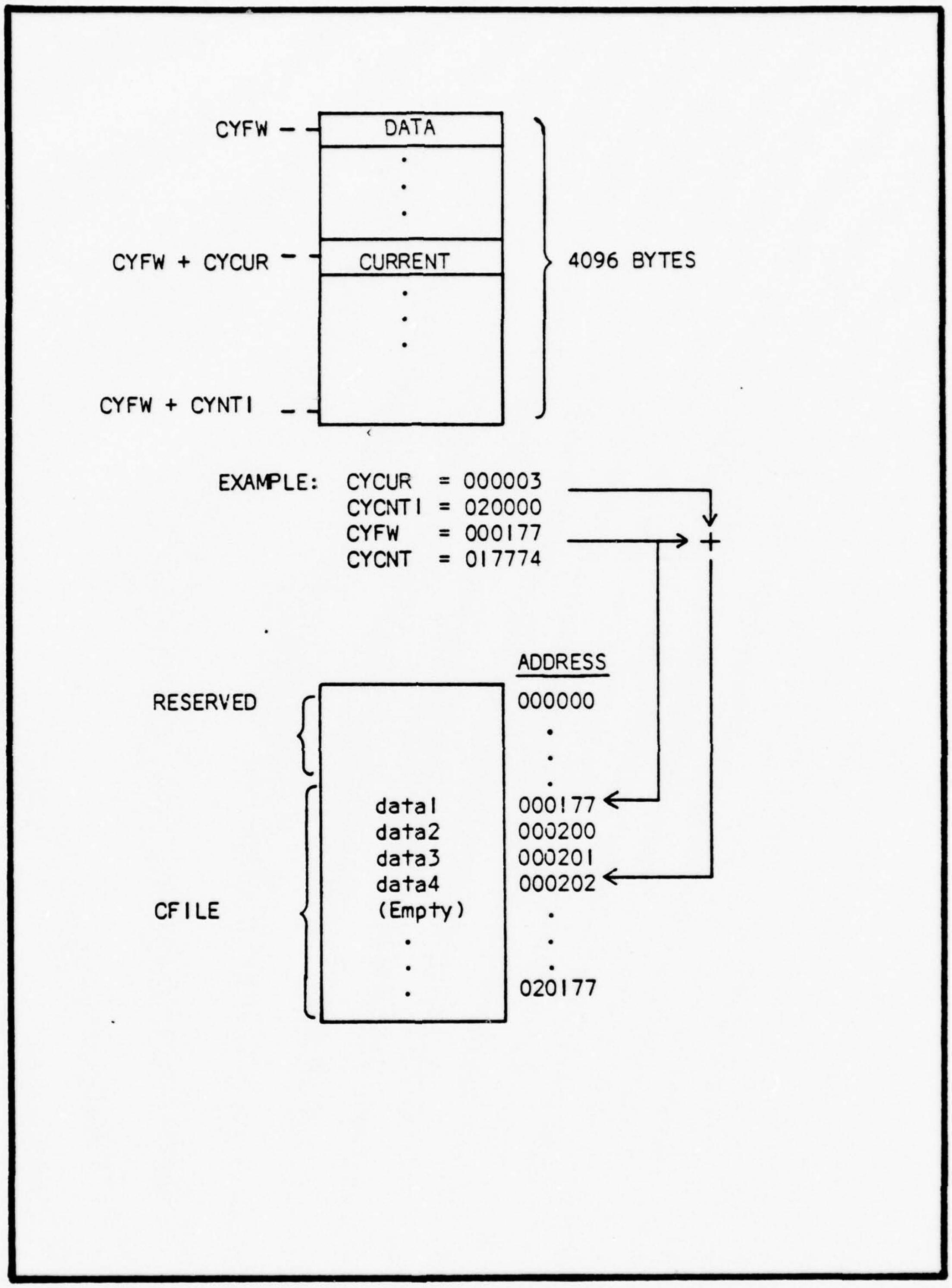


Figure 16 CFILE Input Buffer

When the CYBER CP is executed, the program data base is initialized by reading values stored using BASIC DATA statements. The parameters that are initialized are shown in Table 8. Once the data base has been initialized, the CYBER program then initializes the Altair vectored interrupt system and real time clock. The remaining function of the CYBER program is to continue monitoring the command console (Port 020) for an operator input. The operator inputs are processed as shown in Figure 17 (Sheet 3). A description of each command is shown in Table 4. The User's Manual, Section 3.0 describes the operational procedures for each command. The initialization and termination procedures are also described in the User's Manual. Once the CYBER program is initialized, the user can control the receiving and transmitting of data files between the Altair 8800 computer and the CYBER 74 computer using the commands in Table 4.

Inputs. The primary CYBER inputs are operator commands and the diskette data file requested when a SA command is executed. The data base parameters set and used by the CYBER CP are described in Chapter III.

Outputs. The CYBER CP outputs consists of three types of outputs. The first type of output is the data transferred to the CYBER 74 computer as a result of the TR command. The second type is the system status messages shown in Table 9 that occur as a result of normal user operations. The third class of output is the system error messages shown

Table 8

MNUEMONIC	INITIAL VALUE
M1	10239
M	64
E	66
E1	68
C	64613
C1	84
C2	86
C3	90
C4	92
C5	88
C6	102
C7	93
T1	103
T3	65024
T5	65221
T6	65223
T7	167
T8	171
T9	163
D	0

CYBER Initialization Parameters

Table 9

MESSAGE	SOURCE PROGRAM/SUBROUTINE CYBER/MAIN	SOURCE COMMAND
REQUEST-		N/A
TIME:<hrs>:<min>:<sec>	CYBER/TI	TI
ABORT CYIN	DOSCYB/CONTG	CONTROL/D
DATA MODE	DOSCYB/TELE1	CONTROL/T
TELE MODE	DOSCYB/CONTG	CONTROL/T
CYBER MESSAGE RECEIVED. <hrs>:<min>:<sec>	CYBER/CY	CY
END OF PROGRAM: <hrs>:<min>:<sec>	CYBER/MAIN	BY

CYBERSCP Normal Messages

Table 10

Error Message	Cause	Command
ILLEGAL REQUEST-(com)*	Invalid or misspelled request entered	All
ARGUMENT ERROR-(arg)	Invalid arument entered after valid command keyword	All except CY
ARGUMENT OUT OF CFILE-(arg)	Range of <start> or <stop> address is out of the CFILE buffer	LC
NUMERIC ARGUMENT EXPECTED-(arg)	A number was expected in the argument	LC, TI
<SOURCE> DOES NOT EXIST - (source)	File <source> could not be found	TR, SA, LF
EMBEDDED BLANK IN ARG-(arg)	A blank was found in the <source> or <dest> name	TR, SA, LF
MISSING ARGUMENT-(com)	A required argument parameter was missing	(All except CY)
<DEST> DOES NOT EXIST-(dest)	File <dest> could not be found	TR, SA, LF
<DEST> FILE FULL - (dest)	<dest> file on the diskette is full	SA
BUFFER FULL-(number) BYTES	The input buffer CFILE is full	CY

*( ) indicate variable output conditions

CYBERSCP System Error Messages

Table 10 which result from various illegal operations attempted by the operator.

Program Listing. The CYBER program listing is provided in Appendix E.

TIME Program. The TIME program is written in 8080A machine code and is resident in PROM at locations FE00(HEX) to FFFF(HEX). The primary function of this program is to keep track of the time-of-day and the time-out duration counter. The TIME program is called initially by the CYBER program to turn on the system real-time clock. This clock generates interrupts every 1/60th of a second which cause the TIME program to execute. The TIME program increments the time of day counter, NMB, once for each interrupt. The program uses this count to keep track of seconds, minutes, and hours for the CYBERSCP. In addition to keeping track of the time-of-day counter, NMB, the program also monitors the duration counter, TISET, which is set by the TI command. To do this, TIME maintains a second counter, TICNT, which is incremented in the same fashion as the time-of-day counter. If the value in TICNT is equal to the value set in TISET, TIME will call the subroutine indicated by the TIFLG parameter. Currently the TIFLG parameter may only contain the values zero or one. If it is a zero, then no action will occur. If it is a one then the TIME program will call the TITRP1 subroutine. This subroutine sends the message "HELLO" to the CYBER 74 in order to prevent the CYBER 74 from timing

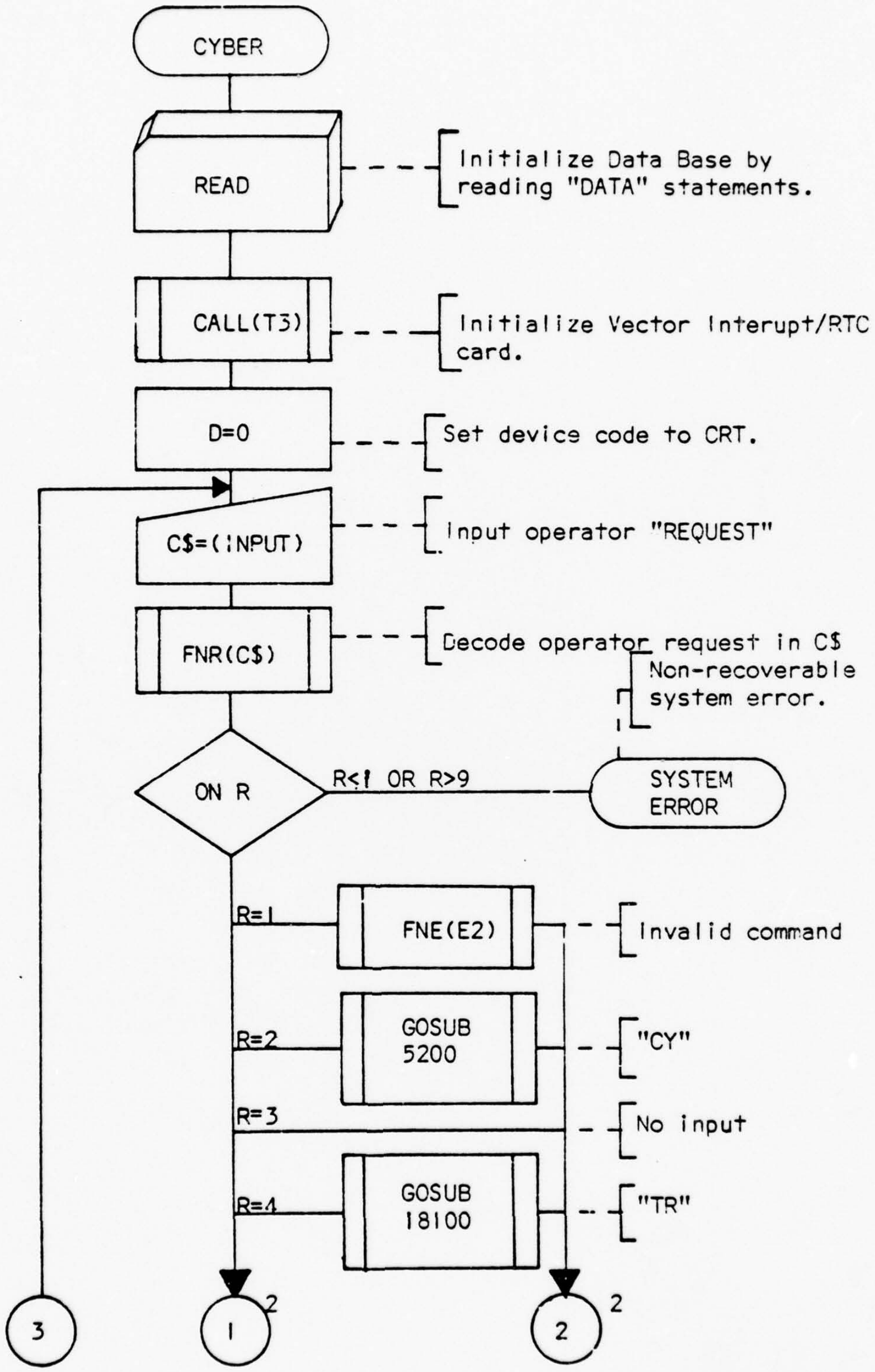


Figure 17 CYBER Flowchart (Sheet 1 of 19)

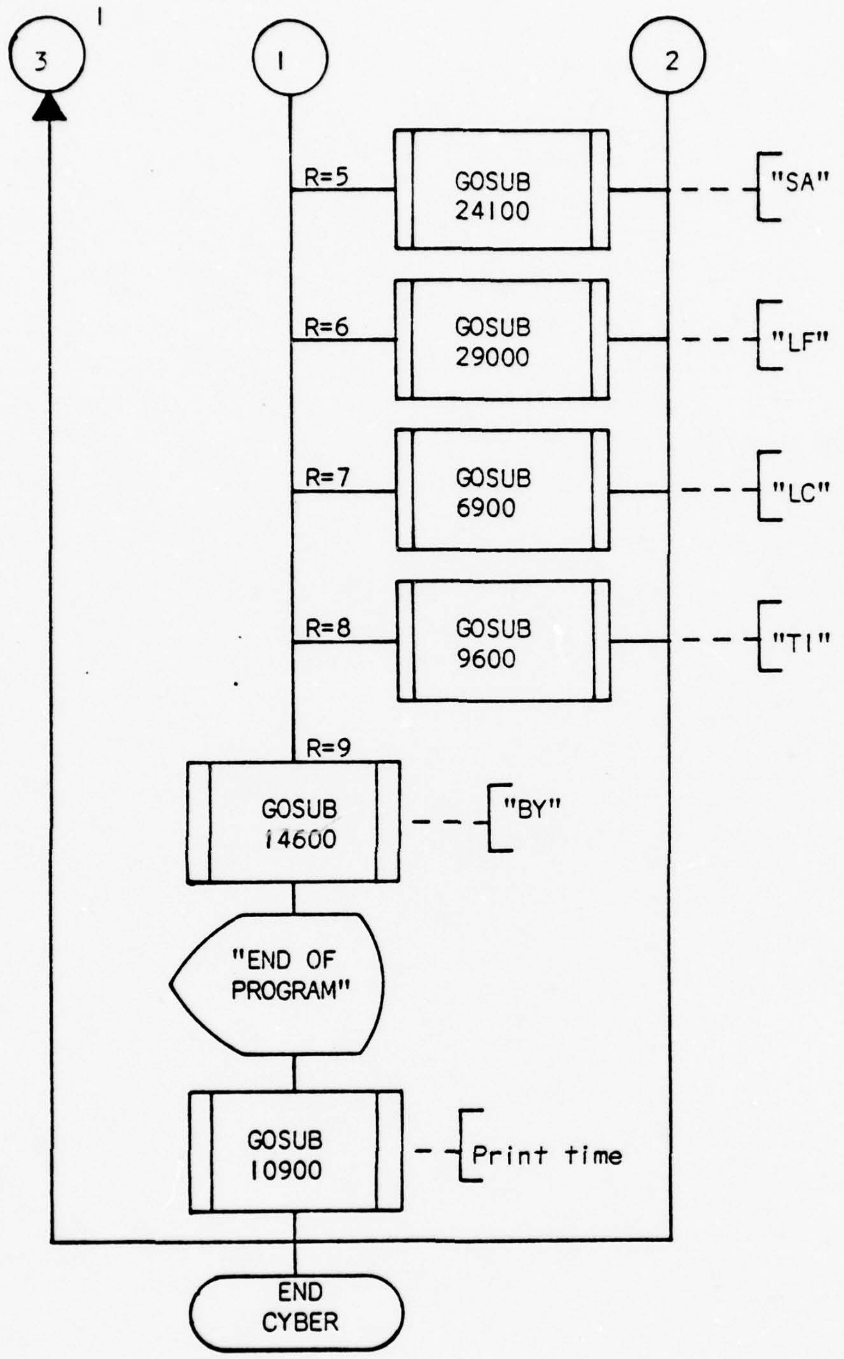


Figure 17 CYBER Flowchart (Sheet 2 of 19)

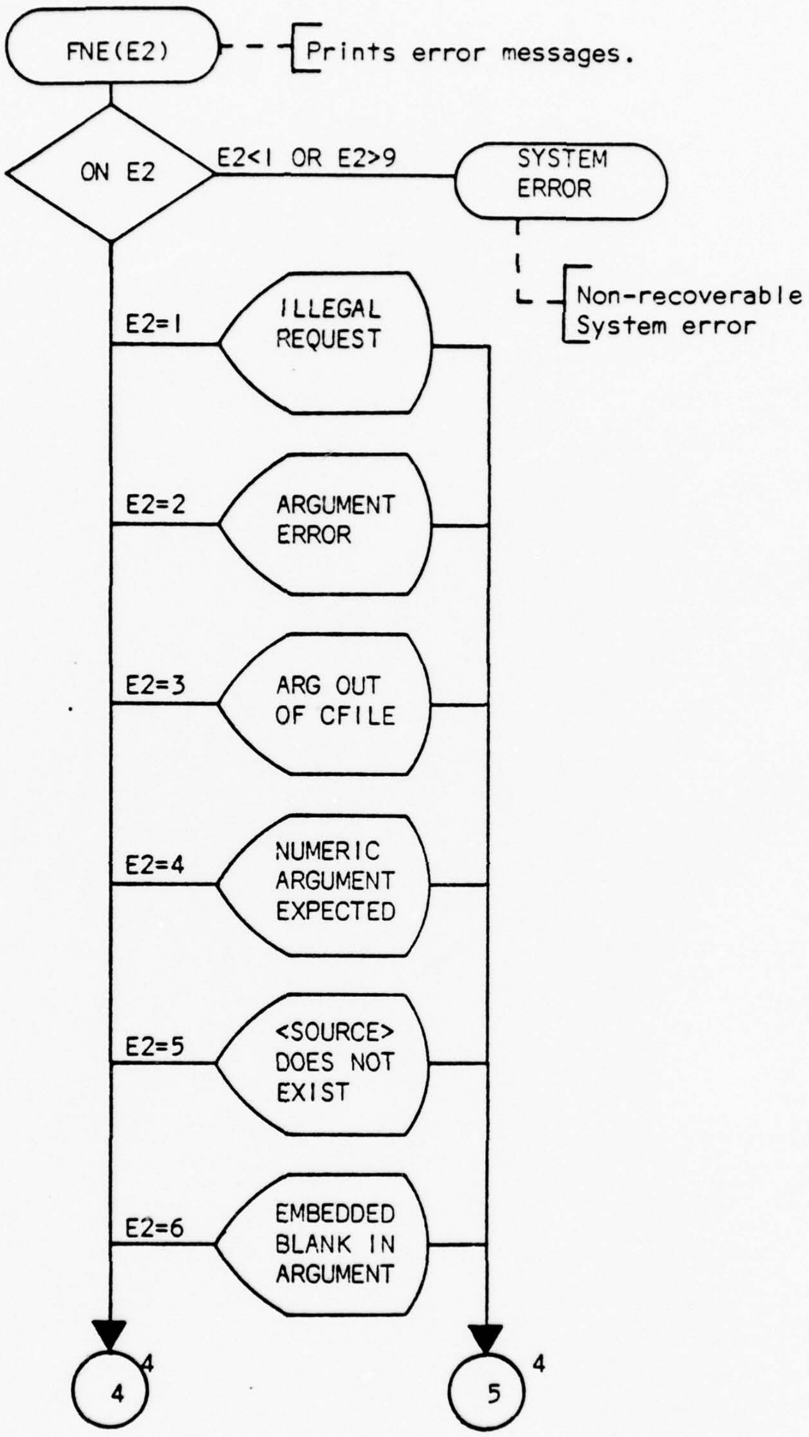


Figure 17 CYBER Flowchart (Sheet 3 of 19)

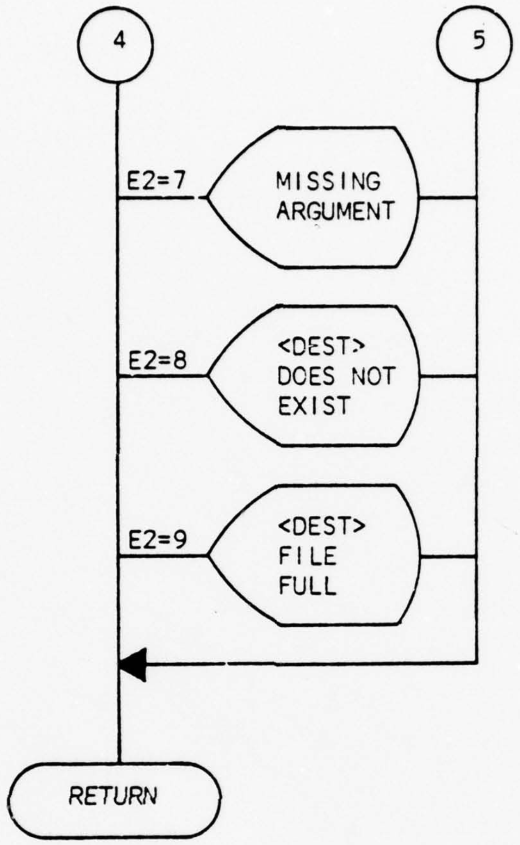


Figure 17 CYBER Flowchart (Sheet 4 of 19)

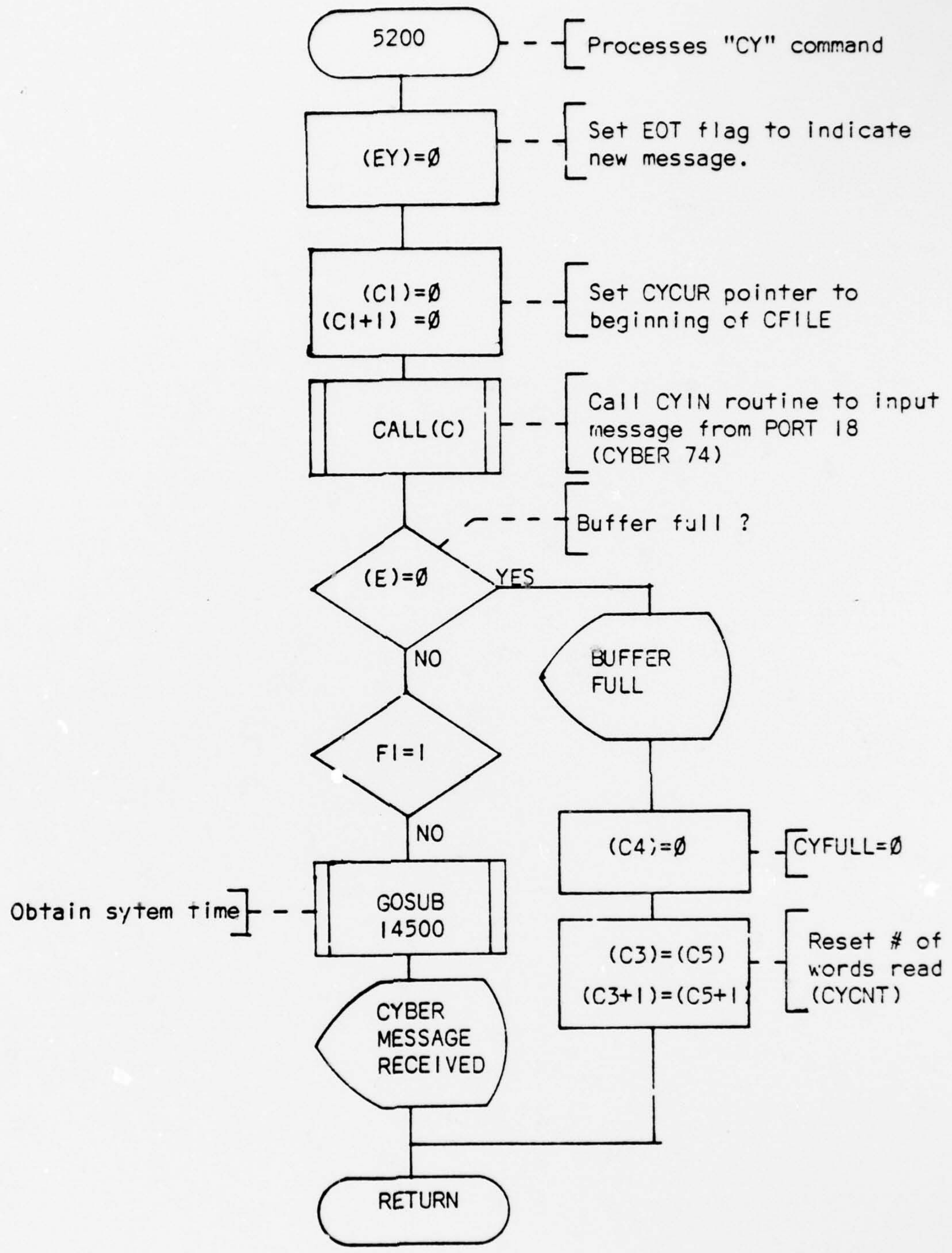


Figure 17 CYBER Flowchart (Sheet 5 of 19)

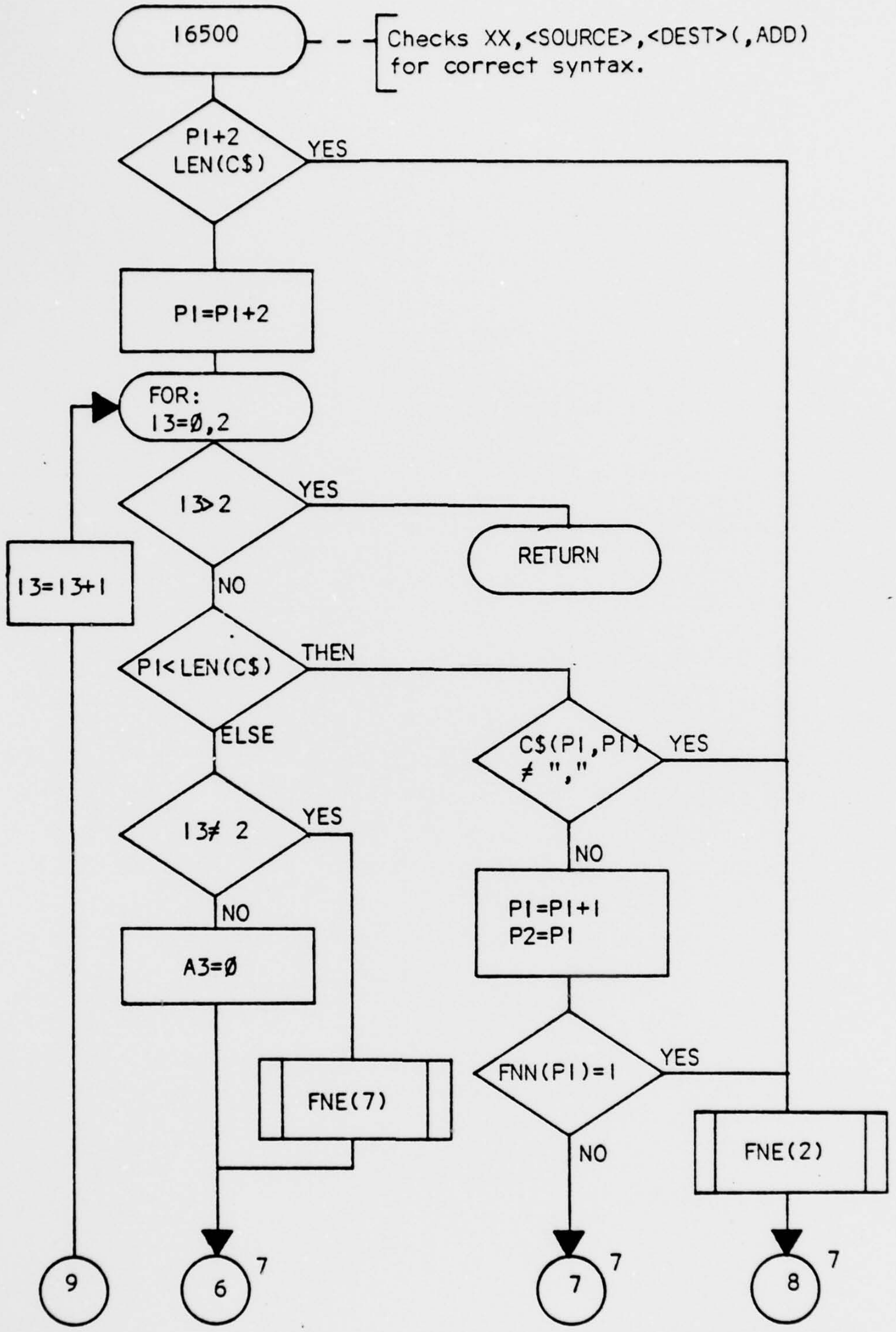


Figure 17 CYBER Flowchart (Sheet 6 of 19)

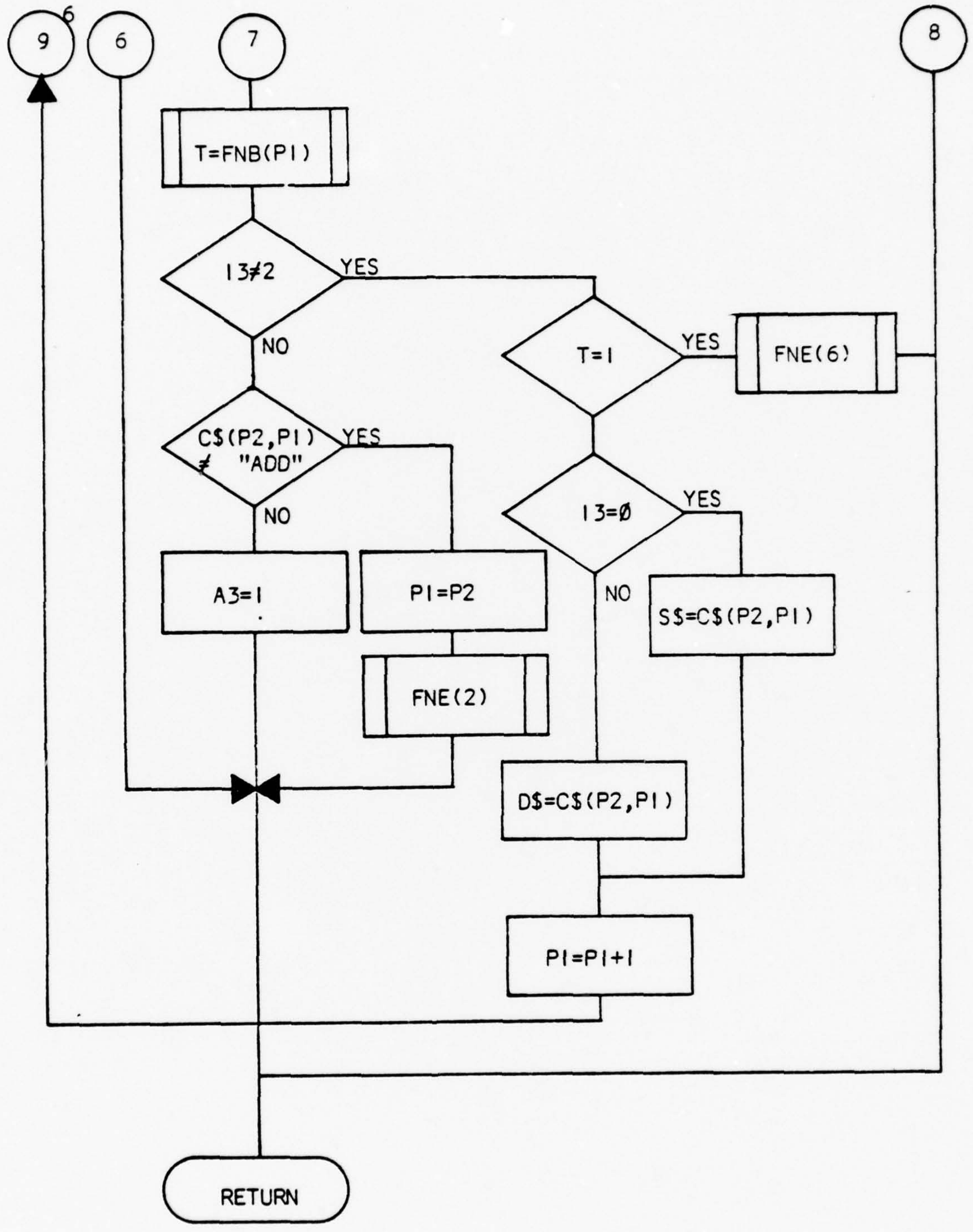


Figure 17 CYBER Flowchart (Sheet 7 of 19)

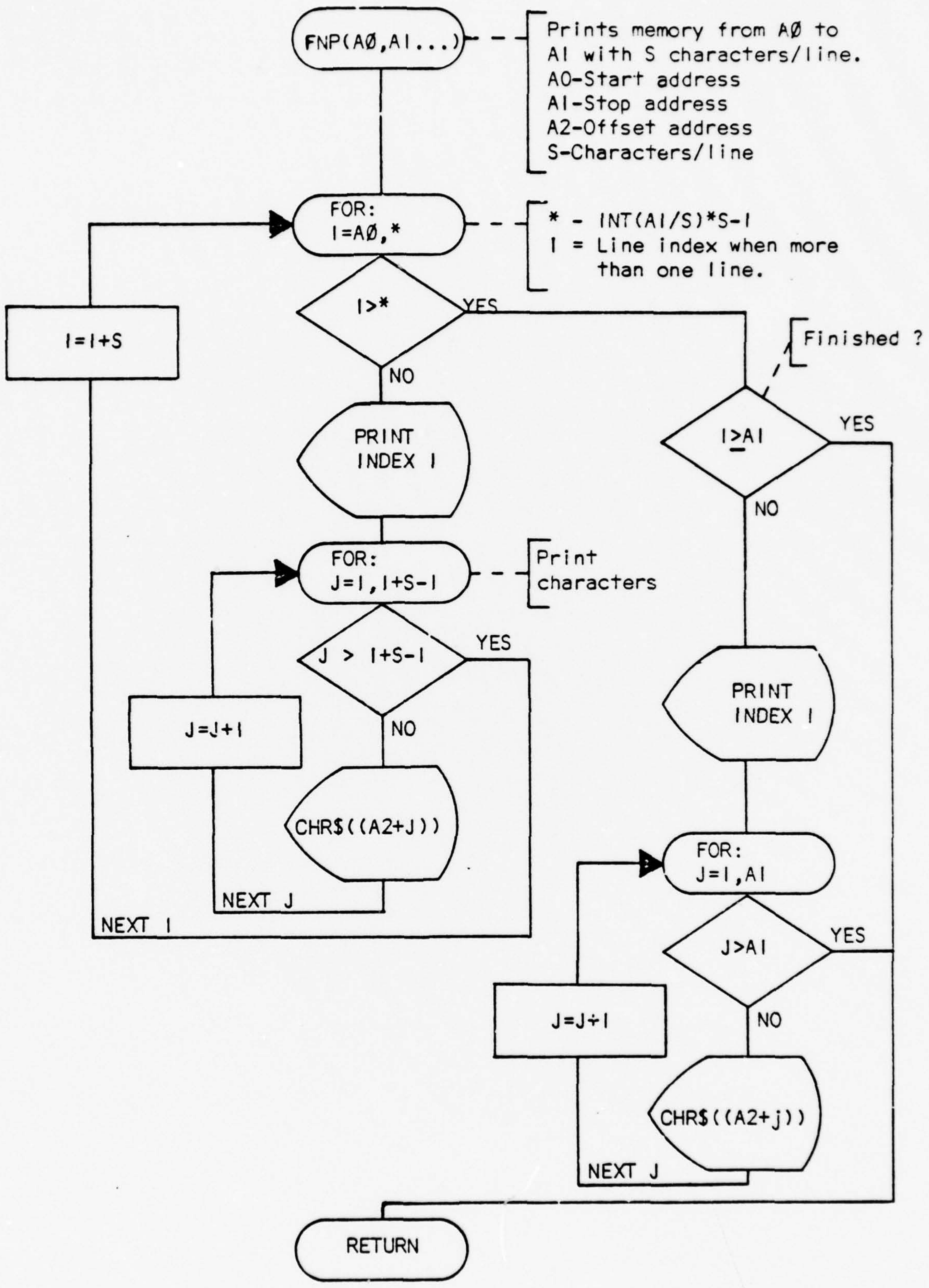


Figure 17 CYBER Flowchart (Sheet 8 of 19)

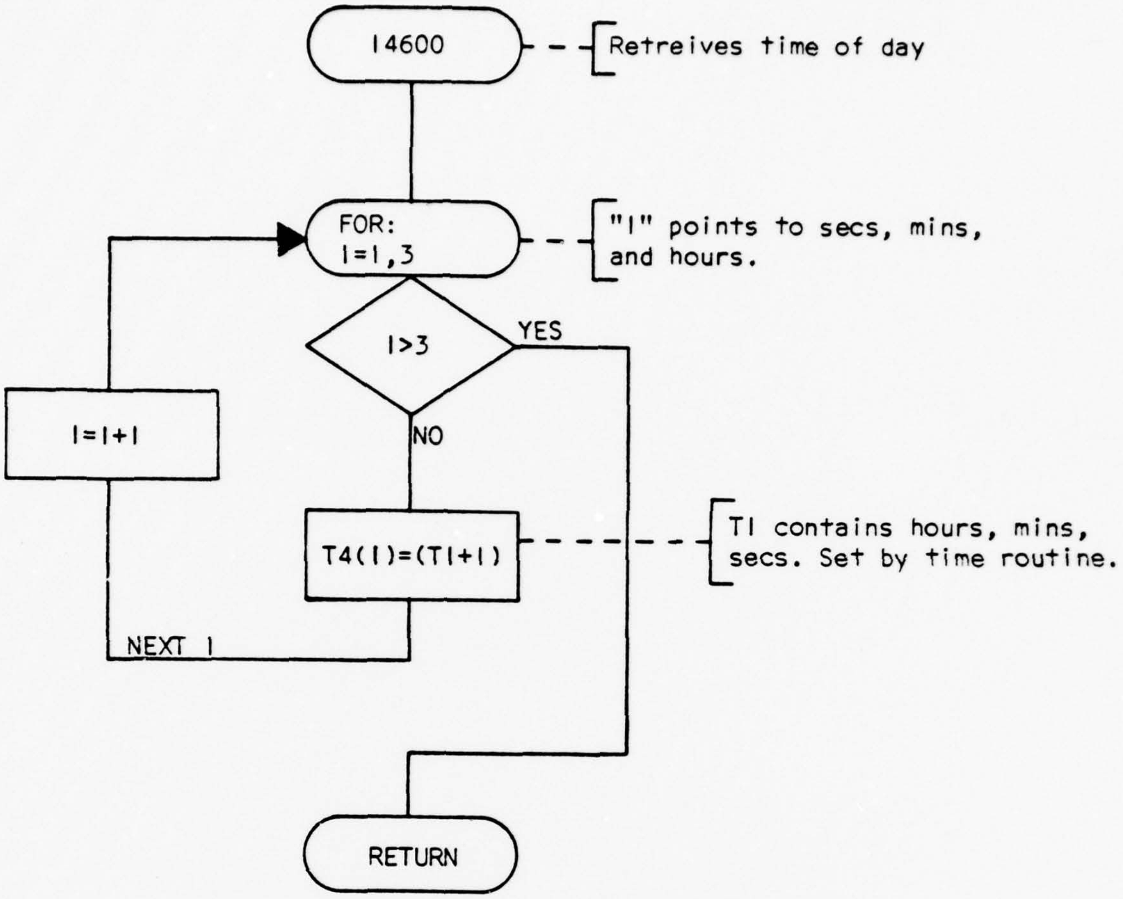

14600
Retreives time of day
FOR:
I=1,3
"I" points to secs, mins, and hours.
I>3
YES
NO
I=I+1
T4(I)=(TI+I)
TI contains hours, mins, secs. Set by time routine.
NEXT I
RETURN

Figure 17 CYBER Flowchart (Sheet 9 of 19)

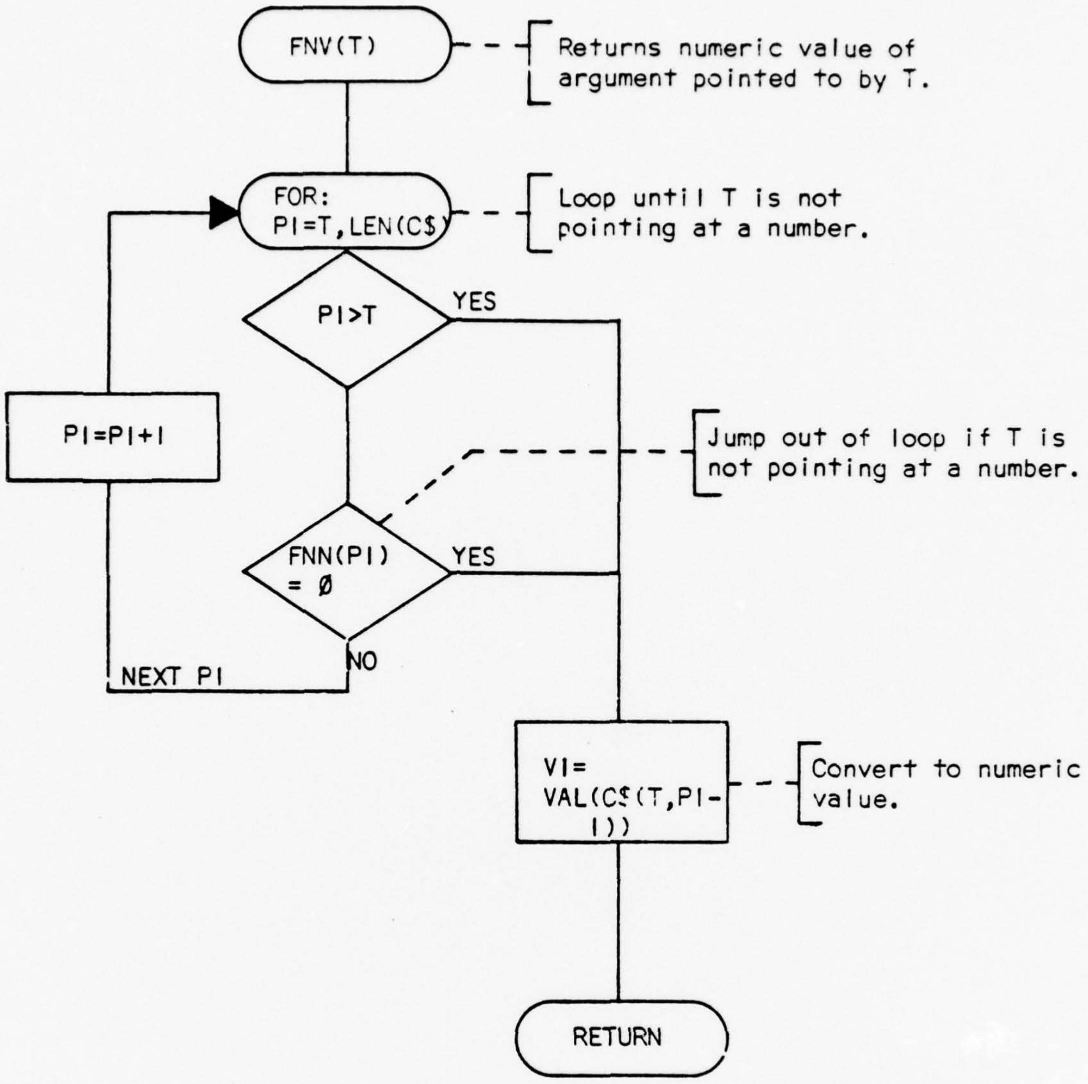

FNV(T)
Returns numeric value of argument pointed to by T.
FOR: PI=T,LEN(C$)
Loop until T is not pointing at a number.
PI>T
YES
PI=PI+1
Jump out of loop if T is not pointing at a number.
FNN(PI) = Ø
YES
NO
NEXT PI
VI= VAL(C$(T,PI-1))
Convert to numeric value.
RETURN

Figure 17 CYBER Flowchart (Sheet 10 of 19)

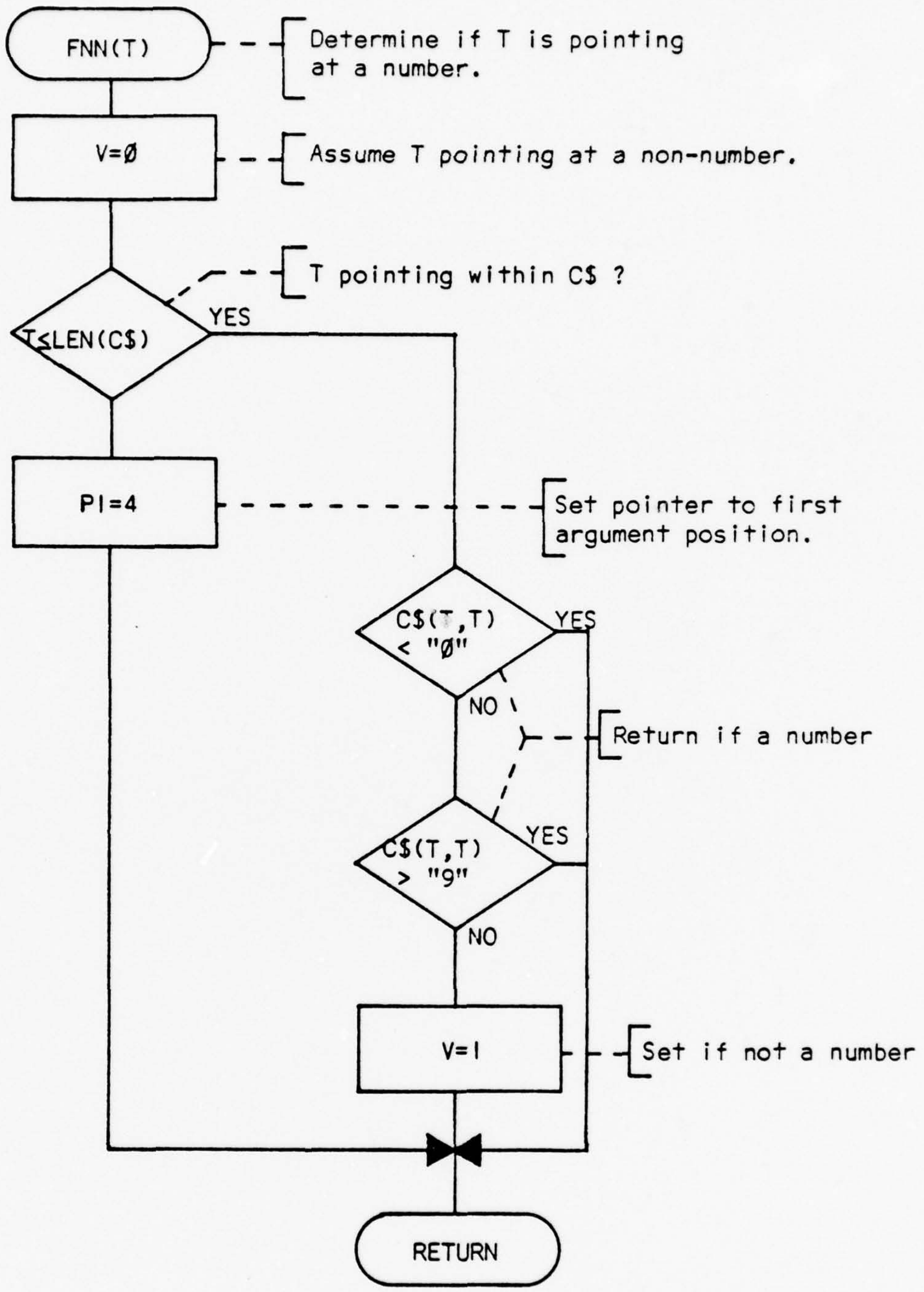


Figure 17 CYBER Flowchart (Sheet 11 of 19)

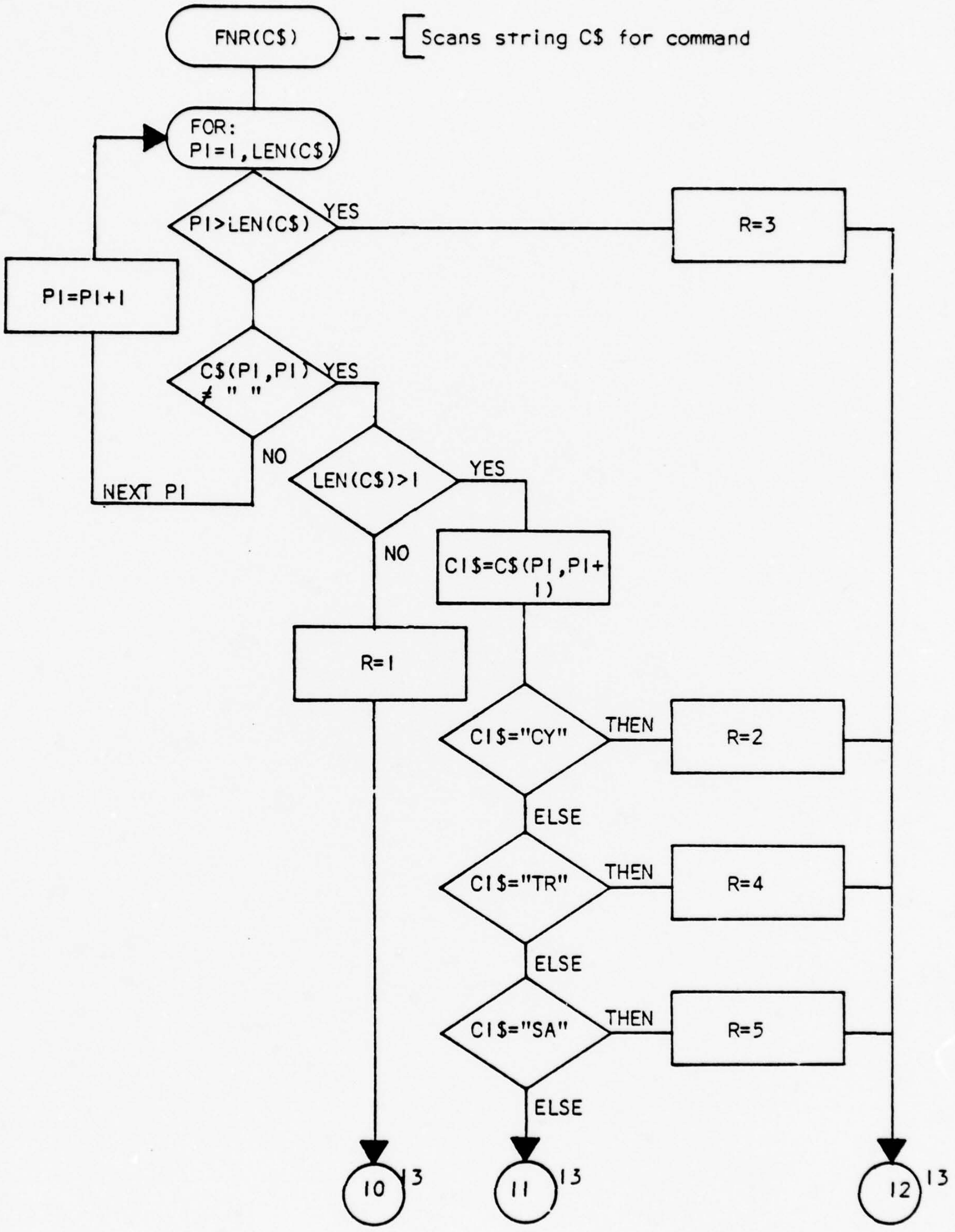


Figure 17 CYBER Flowchart (Sheet 12 of 19)

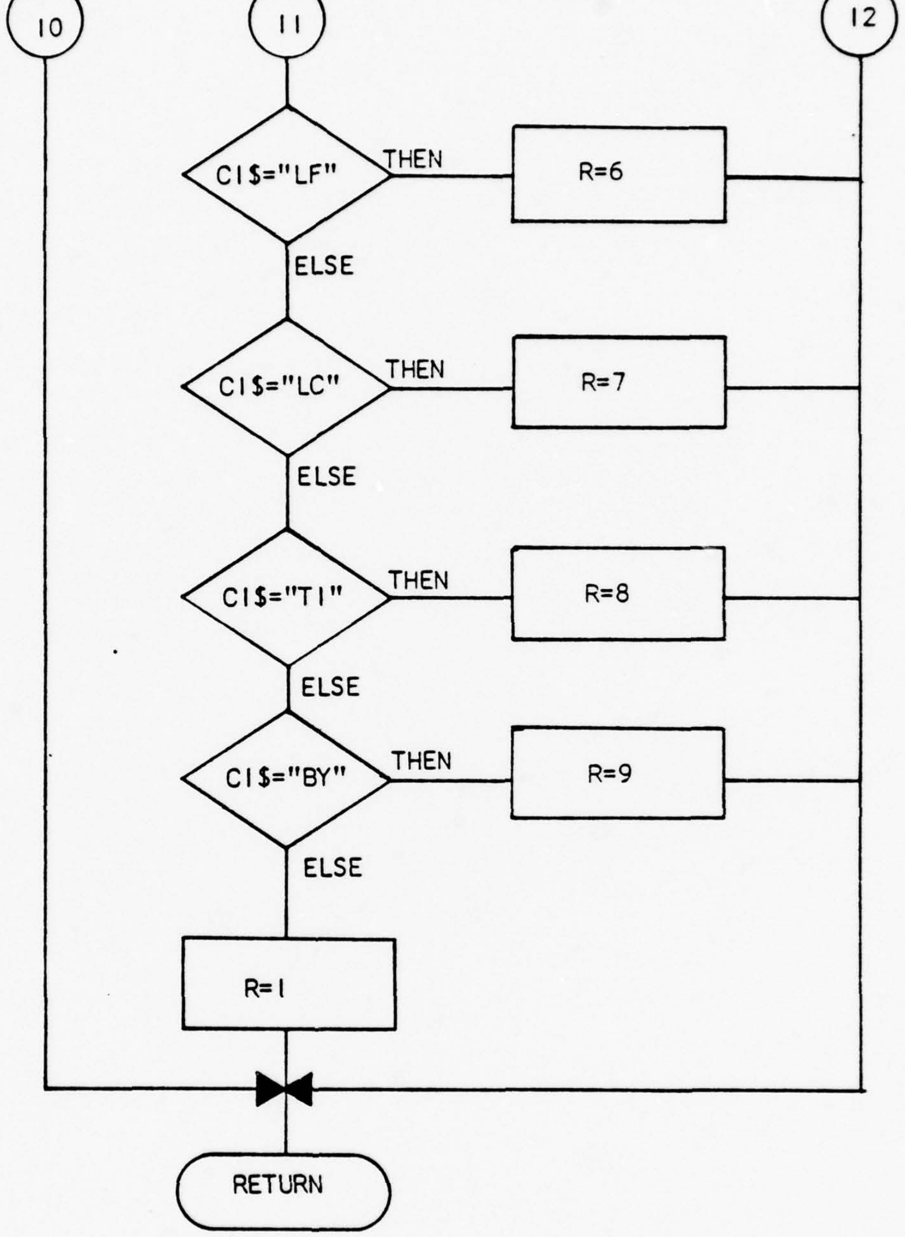


Figure 17 CYBER Flowchart (Sheet 13 of 19)

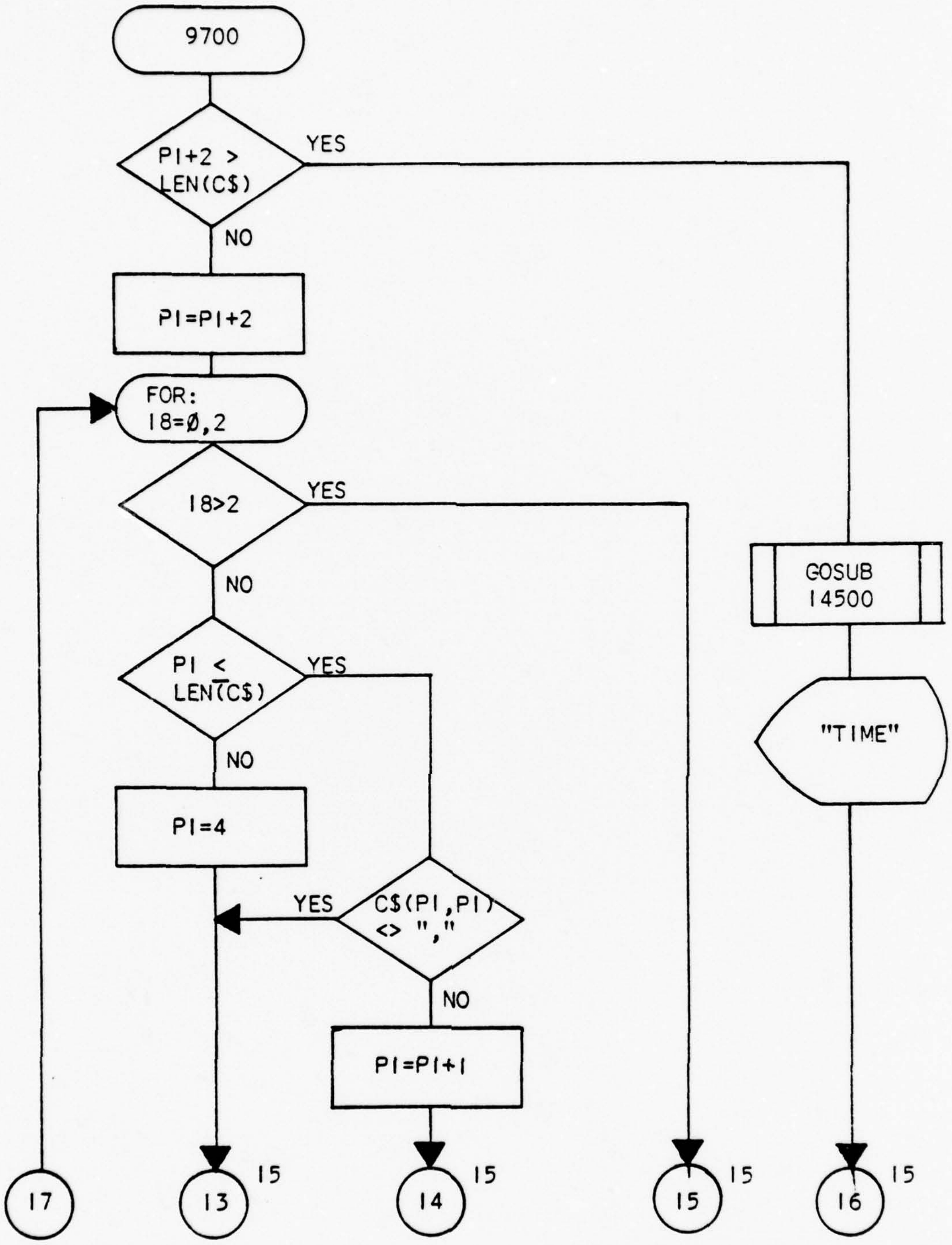


Figure 17 CYBER Flowchart (Sheet 14 of 15)

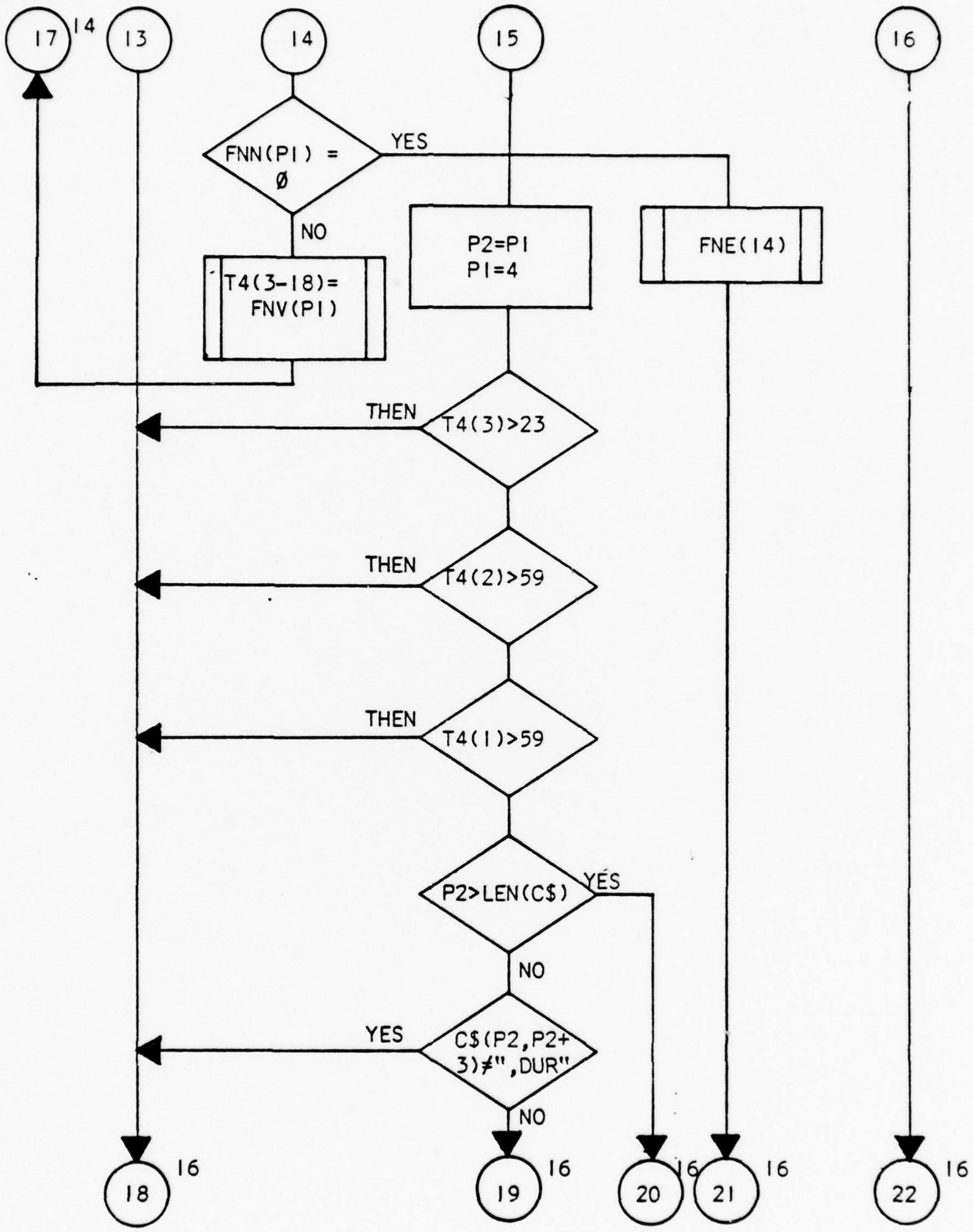


Figure 17 CYBER Flowchart (Sheet 15 of 19)

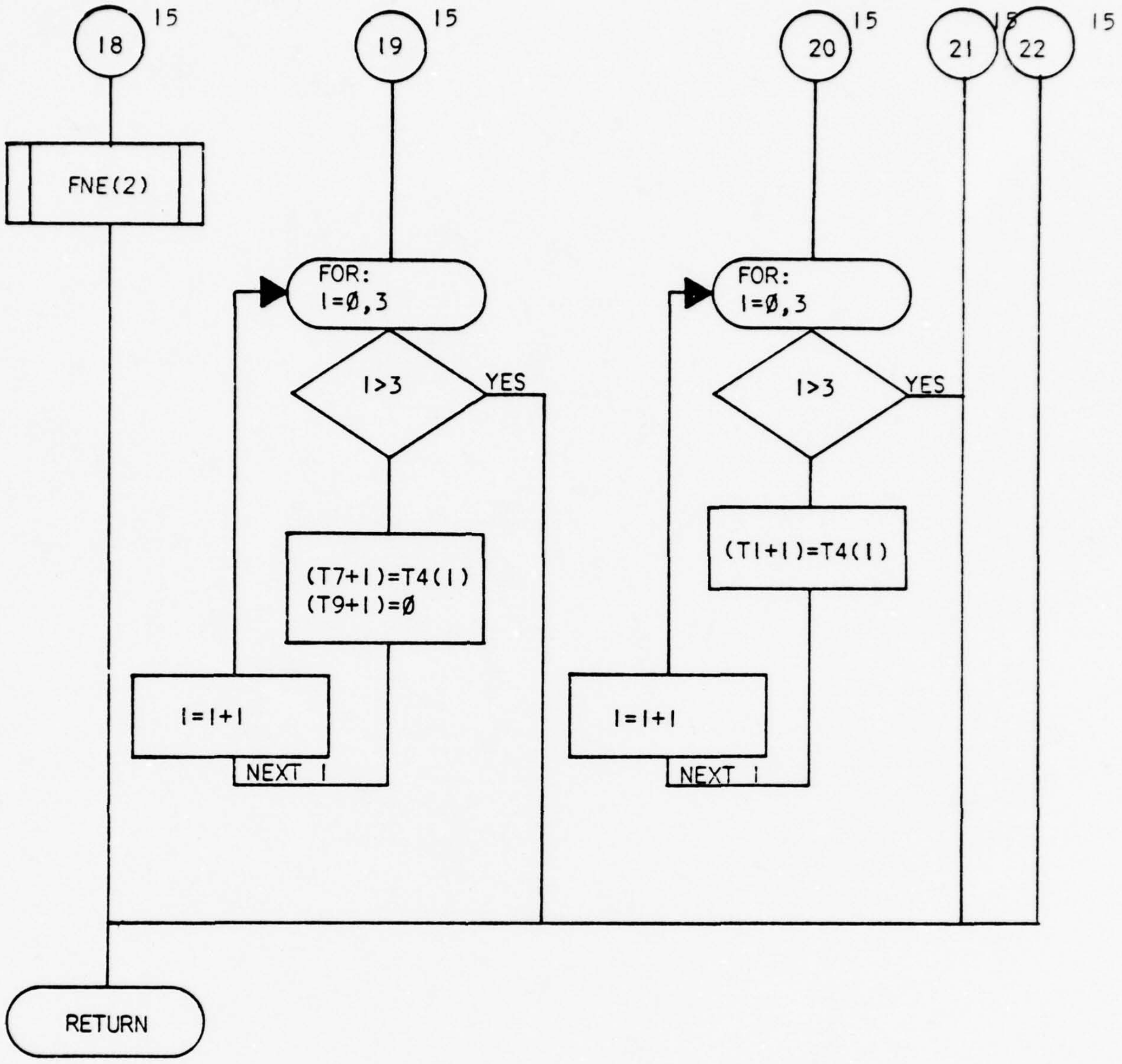


Figure 17 CYBER Flowchart (Sheet 16 of 19)

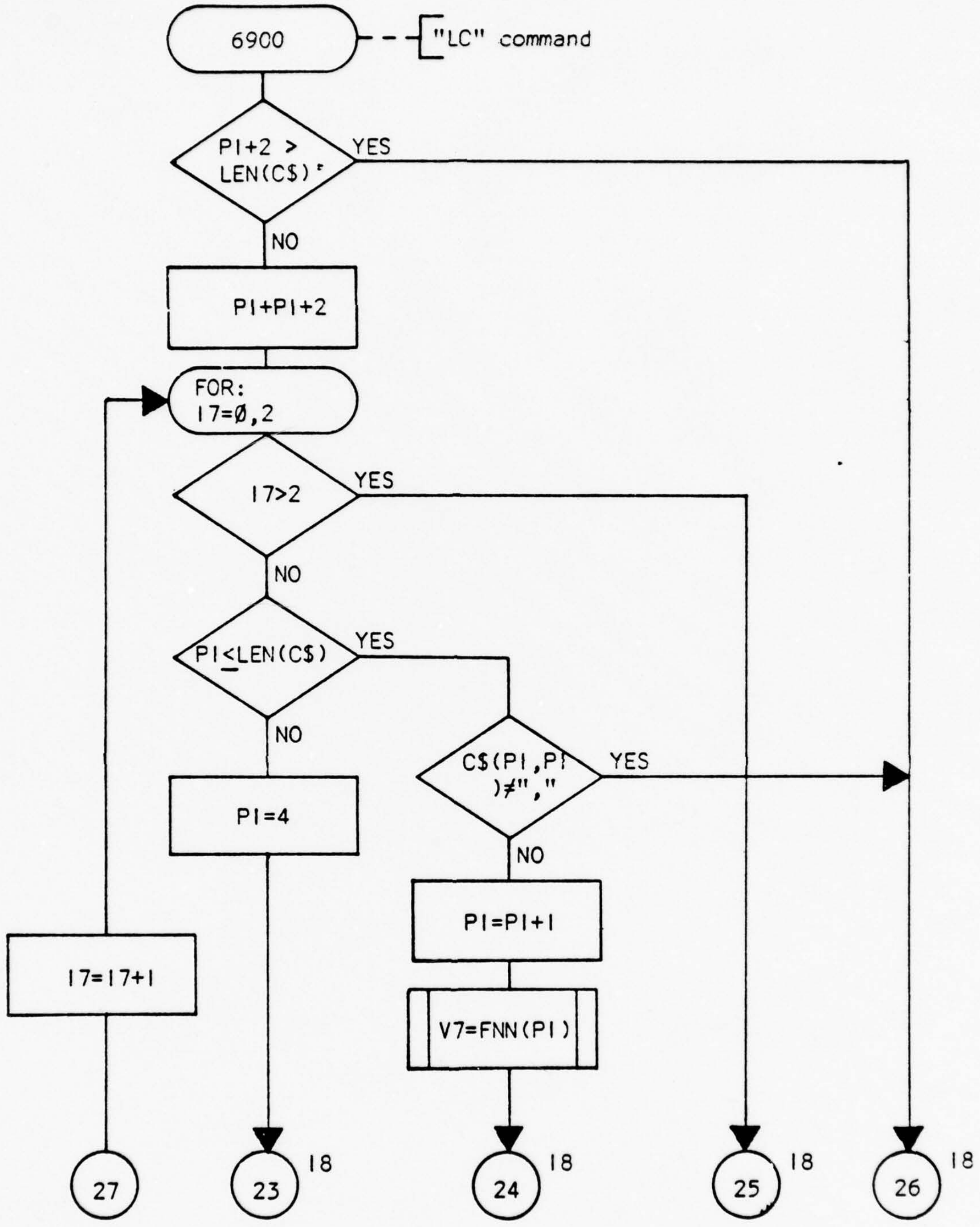


Figure 17 CYBER Flowchart (Sheet 17 of 19)

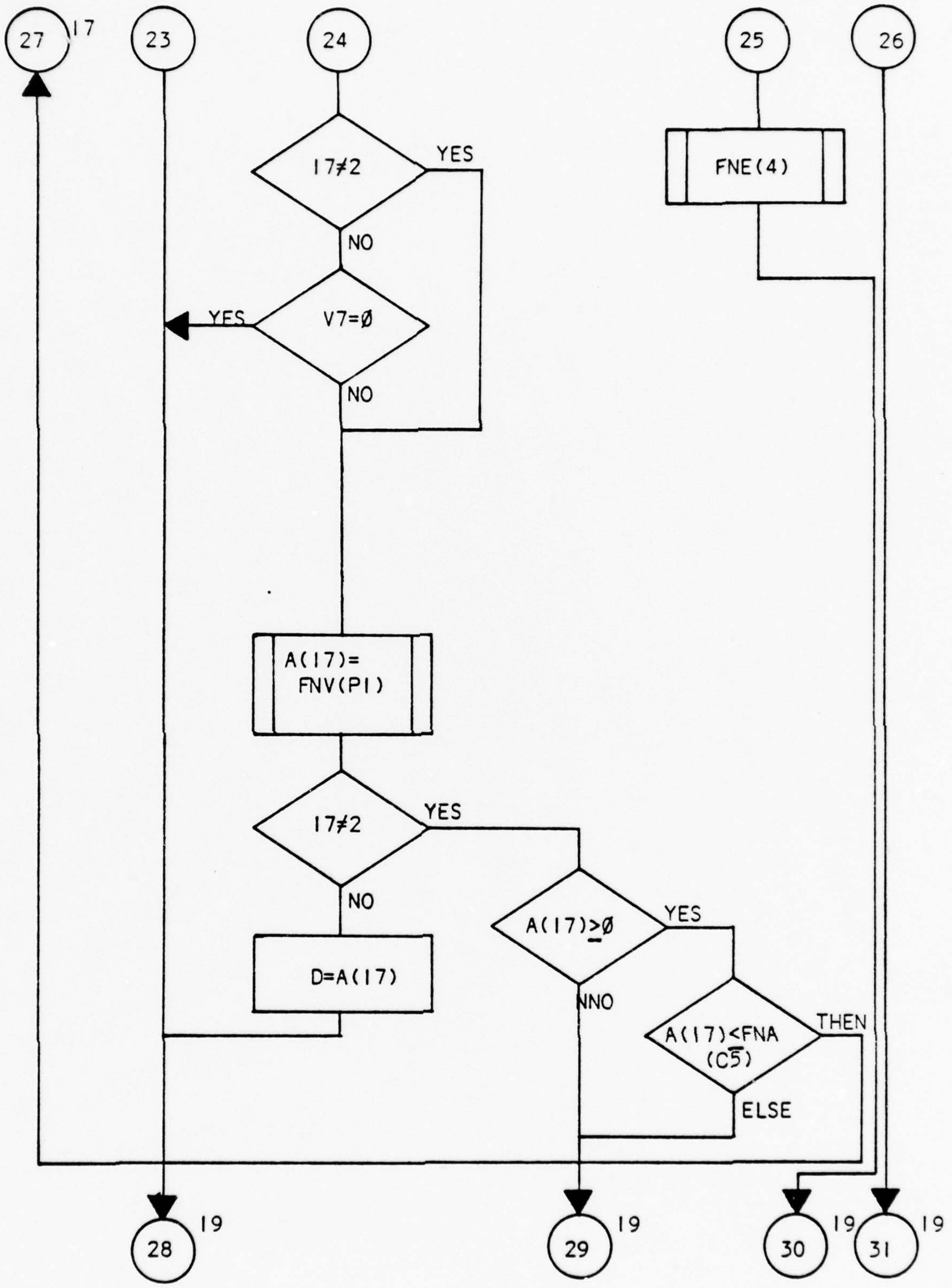


Figure 17 CYBER Flowchart (Sheet 18 of 19)

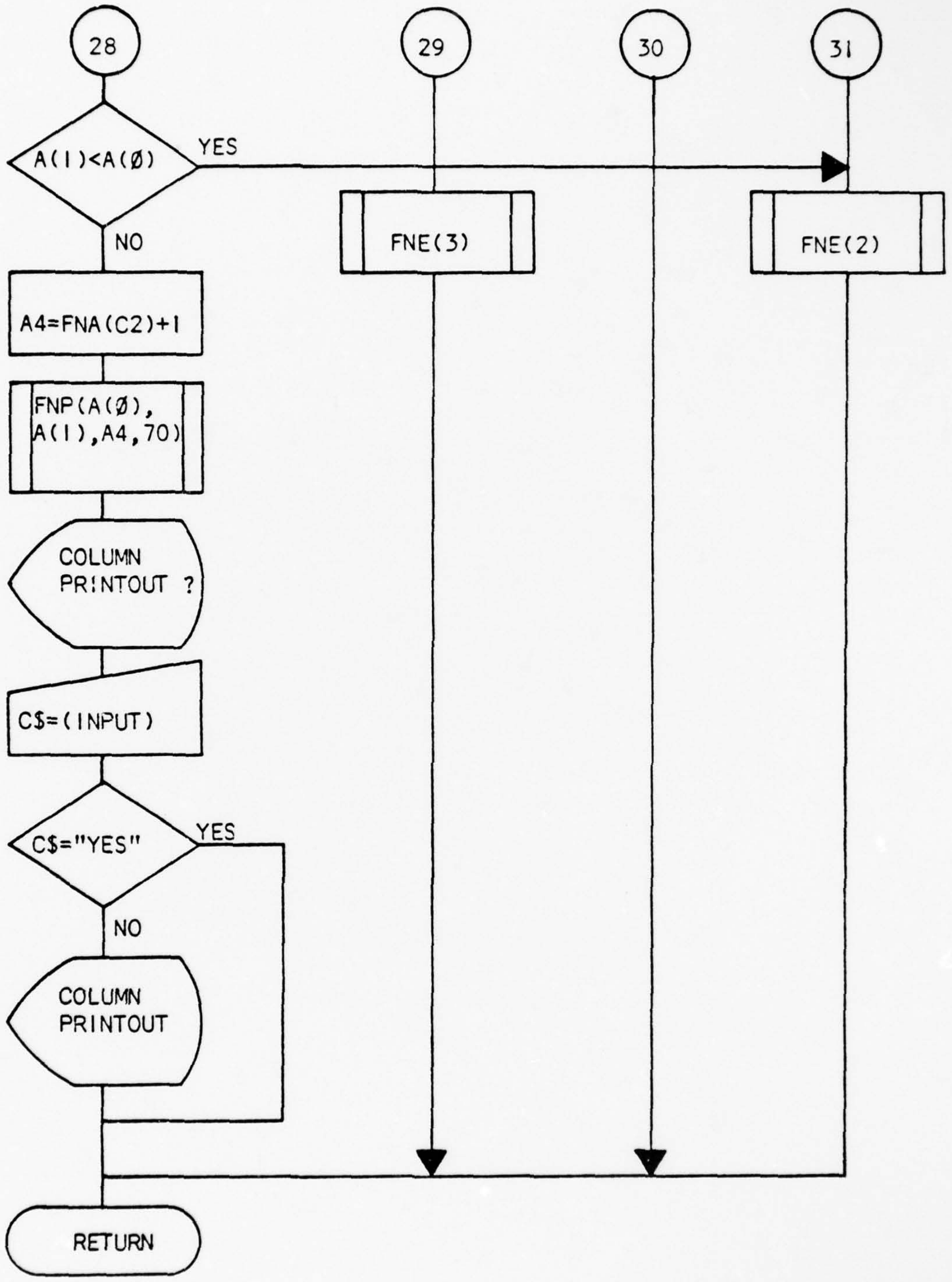


Figure 17 CYBER Flowchart (Sheet 19 of 19)

out due to no operator action and then resets the TICNT counter. If any other value for TIFLG is set, no action will occur. The TIME functional flow is described in Figure 18.

Inputs. The inputs to the TIME program are TISET and TIFLG. TISET contains the length of the duration counter in seconds, minutes, and hours. It is set using the TI command. The TIFLG indicates the subroutine that is called when the TISET duration is expired.

Outputs. The TIME program outputs the time-of-day to the system data base in the NMB parameter. This time is in seconds, minutes, and hours format.

Program Listing. The TIME program listing is in Appendix E.

CYBERSCP Program Summary

This chapter has described the five computer programs that comprise the CYBERSCP system. The detailed flowcharts describe the functional flow of each program. The next chapter describes the data base structure and the specific data base parameters used in the CYBERSCP system.

IX CYBERSCP Data Base

This chapter describes the system data base structure of the CYBERSCP. It describes the general memory partitioning of the Altair computer and then describes the specific data base parameters used by the system.

Memory Partioning

The Altair computer memory used in the CYBERSCP System contains 32k of RAM/ROM memory. This memory is used to store both the computer programs and the data base parameters used by these programs. The memory partioning used is shown in Figure 19.

Data Base Parameters

This section describes the data base parameters associated with the DOSCYB program and the CYBER program. The data base item description includes the location in memory, set/use parameters, initial value and general description. The descriptor, V/C, indicates if the parameter is a variable or constant.

DOSCYB Data Base. The data base parameter associated with the DOSCYB executive are shown in Table 11.

CYBER Data Base. The CYBER program data base is shown in Table 12. The associated assembly code mnuemonies are indicated when appropriate for those parameters that are used by CYBER from the DOSCYB data base.

LOCATION (OCTAL)	CONTENTS	DESCRIPTION
000000 . . . 000057	Blank	Interupt routine linkage
000060 . . . 000070	Level 6 interupt linkage	
000070 . . . 000077	Blank	
000100 . . . 000305	System Data Base	User specified region (Not used by DOSCYB or BASIC)
000305 . . . 017777	CFILE	
020000 . . . 024777	DOSCYB Executive	
025000 . . . 052614	BASIC Interpreter	BASIC Interpreter and source program region
052614 . . . 077777	BASIC source program storage	
100000 . . . 167777	Blank RAM	Additional RAM space

Figure 19 Altair Memory Partioning (1 of 2)

LOCATION (OCTAL)	CONTENTS	DESCRIPTION
170000 . . . 175377	Blank PROM	PROM Region
175400 . . . 176777	CIN3 Program	
177000 . . . 177377	TIME Program	
177400 . . . 177777	Disk (Altair) Bootstrap program	

Figure 19 Altair Memory Partioning (2 of 2)

Summary

This chapter has described the data base structure of the Altair Computer and the specific data base items used by both the DOSCYB executive program and the CYBER program. This description completely describes all of the data base items used by the CYBERSCP.

Table 11

LOC	LENGTH	NAME	SET	USED	INITIAL	V/C	COMMENT
100	1	Mode	BASIC CIN	CIN	0/DOS	V	=0 DATA MODE =1 TELE MODE
101	1	CR		CIN	015/DOS	C	CARRIAGE RETURN
102	1	EOT	CYIN	CYIN	012/DOS	V	=10NO DELIMITER =1 Delimiter
103	1	CRG	CYIN	CYIN	012/DOS	V	=10 for MONITOR =2 for EDITOR
104	1	EDIT	BASIC	CYIN	0/DOS	V	=0 MONITOR,=1 EDITOR
105	10	COM		CYIN	(CR) COM MAND''/DOS	C	Delimiter in MODE
117	1	I	CYIN	CYIN	0/DOS	V	INDEX for COM & EDT
120	1	EDT		CYIN	"(CR).b"/ DOS	C	Delimiter
123	1	EOTC		CYIN	0240	C	EOT check character
124	2	CYCUR·	CYIN	CYIN	0/DOS	V	Points to last word filled in EFILE. STORED(L,H) MUST BE RE-SET BY BASIC
126	2	CYFW	CYIN	CYIN	0001FF	C	Points to start of CFILE
130	2	CYCNTI		CYIN	121111/DOS	C	Initial value for CYCNT. It is equal to the size of CFILE
132	2	CYCNT	CYIN	CYIN	CYCNTI	V	Contains complement of # of words in CFILE
134	1	CYFULL	CYIN BASIC	CYIN BASIC	0/DOS	V	=0 CFILE not full =1 CFILE is full Must be re-set by BASIC

Table 11 (Cont'd)

LOC	LENGTH	NAME	SET	USED	INITIAL	V/C	COMMENT
134	1	CYFULL	CYIN BASIC	CYIN BASIC	0/DOS	V	=0 CFILE not full =1 CFILE is full Must be re-set by BASIC
235	1	PRNNTY	COUT	COUT CYBIN	01	V	contains cout mask
236		PRNTF	CONTQ		1	V	Print flag =0 No print =1 print

DOSCYB Data Base Parameters

Table 12

BASIC NAME	LENGTH	ASSEMB NAME	SET	USED	INIT VAL	V/C	DESCRIPTION
M	1	MODE	CYBER	CYBER	64	C	Address of MODE FLAG
E	1	EOT	CY	CY	66	C	Address of EOT flag
E1	1	EDIT	CY	TR	68	C	Address of EDIT flag
C1	1	CYCUR	CY	TR,SA	84	C	Address of Low order 8-bits of CYCUR
C2	1	CYFW	CY	TR,SA,LC	86	C	Address of Low order 8-bits of CYFW
C3	1	CYCNT	CY		90	C	Address of Low order 8-bits of CYCNT
C4	1	CYFULL	CY, CYBER		92	C	Address of CYFULL flag
C$	72		CYBER			V	Used as input buffer for CRT
R	1		FNR	MAIN		V	contains command code returned by FNR =1 ERROR =6 =2 CY =7 =3 BLANK =8 =4 TR =9 =5 =10
E2	1					V	ERROR FLAG =1 IMPROPER COM. =2 =3 =4 =5 =6
A	1		FNA	"CY"		V	Points to first now blank character in C$
C	1	CYIN	CYBER	CY	64613		Address of CYIN
D	1				0	V	Device Code
C5	1			CY,LC,	88	V	# of COMMANDS

Table 12 (Cont'd)

BASIC NAME	LENGTH	ASSEMB NAME	SET	USED	INIT VAL	V/C	DESCRIPTION
C1$	10		FNR	FNR	88	V	Contains first two characters of command.
T	1			CY TR	0	V	General Purpose Temp.
T1	1		MAIN, TI	CY	103	C	ADDRESS "Time of day" Buffer address
T4	4		8000	TI,CY		V	Time of day (1/60, sec, min,hrs)
T3	1		MAIN	CY	65024	C	Address of TINIT reactive
P1	1		CY	CY LC		V	Points to next character to be scanned
R	1		FRR			C	Contains command COPE
P2	1		TR	TR ERROR			
I3	1		TR	TR			
A3	1		TR	TR		V	=0 NOLAPD>,=1 <ADD>
I7	1		LC	LC		V	Counter
V7	1		LC	LC			
F	1	FLOOK	MAIN	TR	64960	C	Address of FLOOK ROUTINE
C6	1	FPRESF	MAIN	TR	102	V	Address of FPRESF
C7		FNAME	MAIN	TR	93	C	Address of FNAME
V			FNN			V	
V1			FNV				Returned by FNV
I8			TI	TI		V	Counter Index
I			TI FNP LC			V	

Table 12 (Cont'd)

BASIC NAME	LENGTH	ASSEMB NAME	SET	USED	INIT VAL	V/C	DESCRIPTION
F1	1		MAIN	CY	0	V	=0 tells CY to print an EOF message
			TR				=1 mean no messg to be printed
F2	1		FNF	TR	0		Contains active open file #
T$	1000		FNF	FNF	BLANK	V	used as temp.
M1	1		MAIN	SA		C	MAX # of bytes allowed in <DEST> on mini
I4				SA			
T5	1		CYBER	SA,FNL	65094	C	Interrupt enable routine starting address
T6	1		CYBER	SA,FNL	65096	C	Interrupt dis-able routine starting address
T7	1	TISET	CYBER, TI		167	C	Address of TISET
T8	1	TIFLG	CYBER,		171	C	Address of TIFLG
T9	1		TI		163		Address of TICNT

CYBER Data Base Parameters

X Conclusion

This chapter has described the software hierarchy of the CYBERSCP System. It has also described the high level functional flow of the CYBERSCP and the detailed low level functional flow of the system.

The computer programs that comprise the CYBERSCP System were described in terms of their functional flow and data base parameters.

The CYBERSCP System data base was described in terms of the overall Altair memory partioning utilized and the individual data base parameters used in the CYBERSCP System.

The CYBERSCP System does satisfy the main objective which is transmitting and receiving data between the Altair and CYBER 74 computers.

This discussion, then, completely describes the CYBERSCP computer programs.

Bibliography

1. Schwartz, Mischa. Computer-Communications Network Design and Analysis. Englewood Cliffs: Prentice Hall, Inc., 1977.

2. Wood, David C. "A Survey of the Capabilities of 8 Packet Switching Networks," Proceeding of the 1975 Symposium Computer Networks: Trends and Applications. 1-7. New York: Institute of Electrical and Electronic Engineers, Inc., June 1975.

3. McQuillan, J. M., W. r. Crowther, B. P. Cosell, D. C. Walden, and F. E. Heart. "Improvements in the design and performance of the ARPA network," AFIPS Conference Proceedings: 1977 Fall Joint Computer Conference, 41 (Part II). 741-54 (December 1972).

4. McQuillan, John M. Adaptive Routing Algorithms for Distributed Computer Networks. Harvard U. PhD thesis. Bolt Beranek and Newman, Inc., May 1974 (AD-781467).

5. TELENET Communications Corporation. TELENET: The data communications network you can call your own. Advertising brochure. Vienna, Virginia: Telenet Communications Corporation, 1978.

6. Chretion, G. J., W. M. Konig, and J. H. Rech. "The SITA Network, Summary Description," Computer-Communication Networks Conference. Brighton, U.K: University of Sussex, Sept. 1973.

7. Forsdick, Harry C., Richard E. Schantz, and Robert H. Thomas. "Operating Systems for Computer Networks" Computer, II: 48-50 (January 78).

8. Schwager, Andre' O. "The Hewlette-Packard Distributed System Network, "Hewlette-Packard Journal: 2-6 (March 1978).

9. Thomas, R. H. "A Resource Sharing Executive for the Arpanet," AFIPS Conference Proceedings, 1973 SJCC, 42. 155-163 (1973).

10. Crocker, S. D. "The National Software Works: A New Method for Providing Software Development Tools Using the Arpanet," Consiglio Nazional delle Richerche Instituto Di Elaborazione Della Informazione Meeting on 20 Years of Computer Science presentation. Pisa, Italy, June 1975.

Bibliography Cont'd

11. BSR X3.66, X3534-589. Proposed American National Standard for Advanced Data Communication Control Procedures (ADCCP), Seventh Draft. American National Standards Institute, 14 Dec 1977.

12. GA27-3093-1. IBM Synchronous Data Link Control General Information. IBM File no. GENL-09. Research Triangle Park, North Carolina: IBM System Communications Division, Publication Center, 1974.

13. GA27-3004-2. General Information - Binary Synchronous Communications. IBM File no. TP-09. Research Triangle Park, North Carolina: IBM System Communications Division, Publication Center, 1970.

14. Pyke, T. N., Jr. and R. P. Blanc. "Computer Networking Technology-A State of the Art Review" Computer Networks: A Tutorial (1976 Revision). New York: Institute of Electrical and Electronics Engineers, Inc., 1976.

15. Balkovic, M., H. Klancer, S. Klare and W. McGrather. "High-Speed Voice band Data Transmission Performance on the Switched Telecommunications Network," Bell System Journal, 50: (April 1971).

16. Gray, James P. "Line Control Procedures," Computer Networks: A Tutorial (1976 Revision). New York: Institute of Electrical and Electronics Engineers, Inc. 1976.

17. Enslow, Philip H., Jr. "What Is a 'Distributed' Data Processing System?" Computer, 11: 12-21 (January 1978).

18. Stefferud, Einar, David L. Grobstein, and Ronald P. Uhlig. "Wholesale/Retail Specialization in Resource Sharing Networks" Computer Networks: A Tutorial (1976 Revision). New York: Institute of Electrical and Electronics Engineers, Inc., 1976.

19. Eckhouse, Richard H, Jr. and John A. Stankovic. "Issues in Distributed Processing-An Overview of Two Workshops," Computer, 11: 22-26 (January 1978).

20. Ornstein, S. M., F. Heart, W. Crowther, H. Rising, S. Russell, and A. Michael. "The Terminal IMP for the ARPA Computer Network," Proceedings, SJCC, 1972, pp. 243-254.

Bibliography Cont'd

21. Fultz, G. F. and L. Kleinrock. "Adaptive Routing Techniques for Store-and-Forward Computer - Communications Networks," Proc. IEEE International Conference on Communications. Montreal: June 1971, pp. 39-1 to 39-8.

22. Rudin, H. "On Routing and 'Delta Routing': A Taxonomy and Performance Comparison of Techniques for Packet-Switched Networks," IEEE Trans. on Communications, COM-24, 2. 43-59. Jan 1976.

23. Brandt, G. J. and G. J. Chretien, "Methods to Control and Operate a Message-Switching Network," Proc. Symposium on Computer-Communications Networks and Telegraphic. New York: Polytechnic Institute of Brooklyn, 1972, pp. 263-276.

24. Warshall, S. "A Theorem in Boolean Matrices," JACM, 9: (1962), pp. 11-12.

25. Floyd, R. W. "Algorithm 97, Shortest Path," CACM, 5: (June 1962), p. 345.

26. Coffman, Edward G., Jr. and Peter J. Denning. Operating Systems Theory. Englewood Cliffs: Prentice-Hall, Inc., 1973.

27. Heart, F. E., R. E. Kahn, S. M. Ornstein, W. R. Crowther, and D. C. Walden. "The interface message processor for the ARPA computer network," AFIPS Conference Proceeding: 1970 Fall Joint Computer Conference, 36. 551-567 (May 1970).

28. MITS. Vector Interrupt/Real Time Clock. 88-VI and RTC Manual. Albuquerque, New Mexico: MITS, February 1976.

29. Kleinrock, L. "Communication Nets: Stochastic Message Flow and Delay," New York: Dover Publications, 1964.

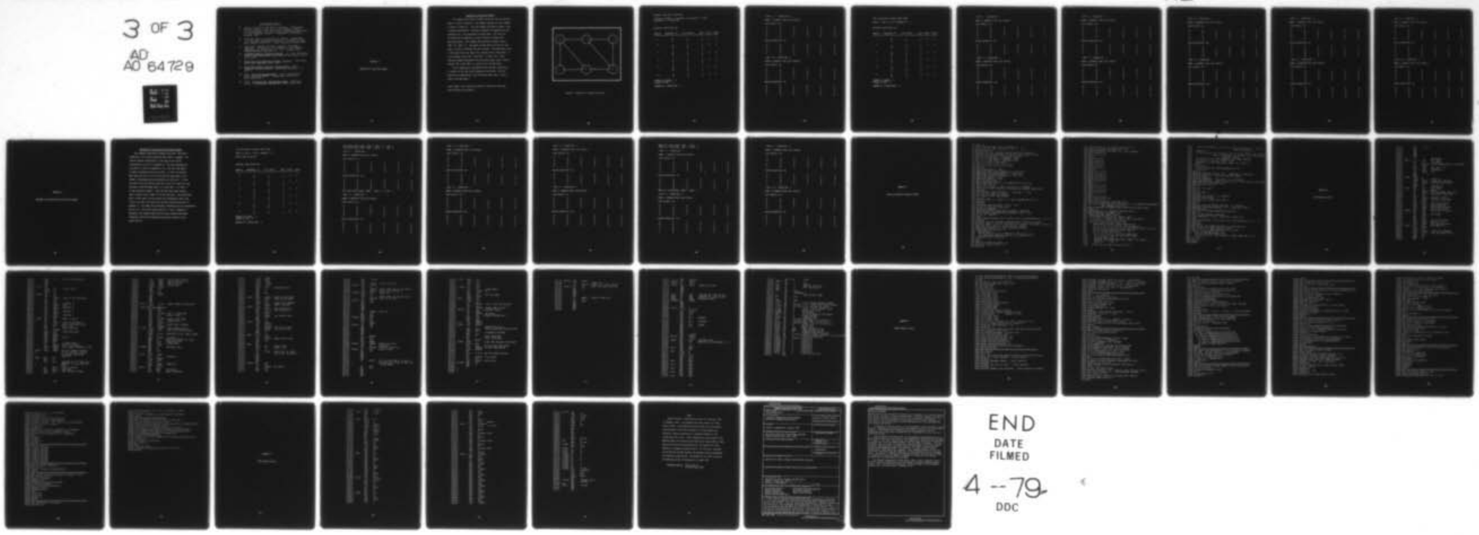
30. Kleinrock, L. "Performance Models and Measurement of the ARPA Computer Network," Proceedings of the International Symposium on the Design and Application of Interactive Computer Systems. England; Brunel University, May 1972.

31. Metcalf, R. M. Packet Communication. Harvard University PhD Thesis. Harvard University, 1973 (MIT MAC TR-114, December 1973).

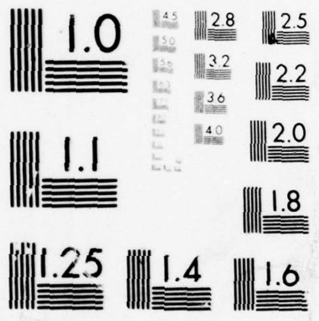
32. Neuman, A. J. et. al., "A Technical Guide to Computer-Communication Interface Standards," NBS Technical Note 843. Aug. 1974.

AD-A064 729 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO SCH--ETC F/G 17/2
ELECTRICAL ENGINEERING DIGITAL DESIGN LABORATORY COMMUNICATIONS--ETC(U)
DEC 78 D L RAVENSCROFT
UNCLASSIFIED AFIT/GCS/EE/78-16-PT-1 NL

3 OF 3
AD A064729



END
DATE
FILMED
4--79
DDC


1.0
4.5
5.0
5.6
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

33. Watson, Richard W. and John G. Fletcher. "End-to-End Protocol Design Issues for a Local Computer network: The Δt Mechanism," 2nd University of Minnesota Conference on Local Computer Networks. Minneapolis, Minnesota, October, 1977.

34. Fletcher, John G. and Richard W. Watson. "Mechanisms for a Reliable Timer-Based Protocol," Symposium on Computer Network Protocols. Liege, Belgium, February 1978.

35. Kleinrock, Leonard, William E. Naylor, and Holger Opderbeck. "A Study of Line Overhead in the Arpanet," Communications of the ACM, 19. 3-6. January 1976.

36. INTERCOM VERSION 4 Reference Manual. St. Paul, Minnesota: Control Data Corporation, Publications and Graphics Division, 1977.

37. North Star Disk Operating System, Version 2. DOS manual. North Star Computers, Inc., 1977.

38. North Star BASIC, Version6, Version6-FPB. BASIC lanugage reference manual. North Star Computers, Inc. 1977.

39. MITS. 88-2 SIO Documentation. Serial Input/Output Board Reference Manual. Albuquerque, New Mexico: MITS, March 1977.

40. MITS. 88-4 Parallel Input/Output Board. Reference Manual. Albuquerque, New Mexico: MITS, March 1977.

Appendix A

Reachability Algorithm Example

Reachability Algorithm Example

This appendix describes an example using the routing algorithm shown in Figure 5 (page 49). The network topology for this example is shown in Figure A-1. The cost between each node is shown in the " Network Configuration " printout located at the beginning of the simulation run. Two simulation runs were made. The first run was made with no anomolies in order to obtain a steady state routing solution. This steady state solution is shown in Run 1, TIME = 18, label (1). The second run was made with the cost from node 2 to node 3 increased from one to three. The significant point is that node 2 must now search for a better path to node three. It first searches using line 1 (See TIME = 1, label (2)). After receiving updated information from the other nodes, node 2 finally chooses line 2 (see TIME = 7, lable (3)) as the best path.

Time is measured by incrementing time with each ocurrance of a transfer of the cost matrix between any two nodes. The cost matrices are transferred in the following order: node 1, node 2, node 3, and then node 4.

(Note- Nodes 1 and 4 must also search for new paths since they were previously using node 2).

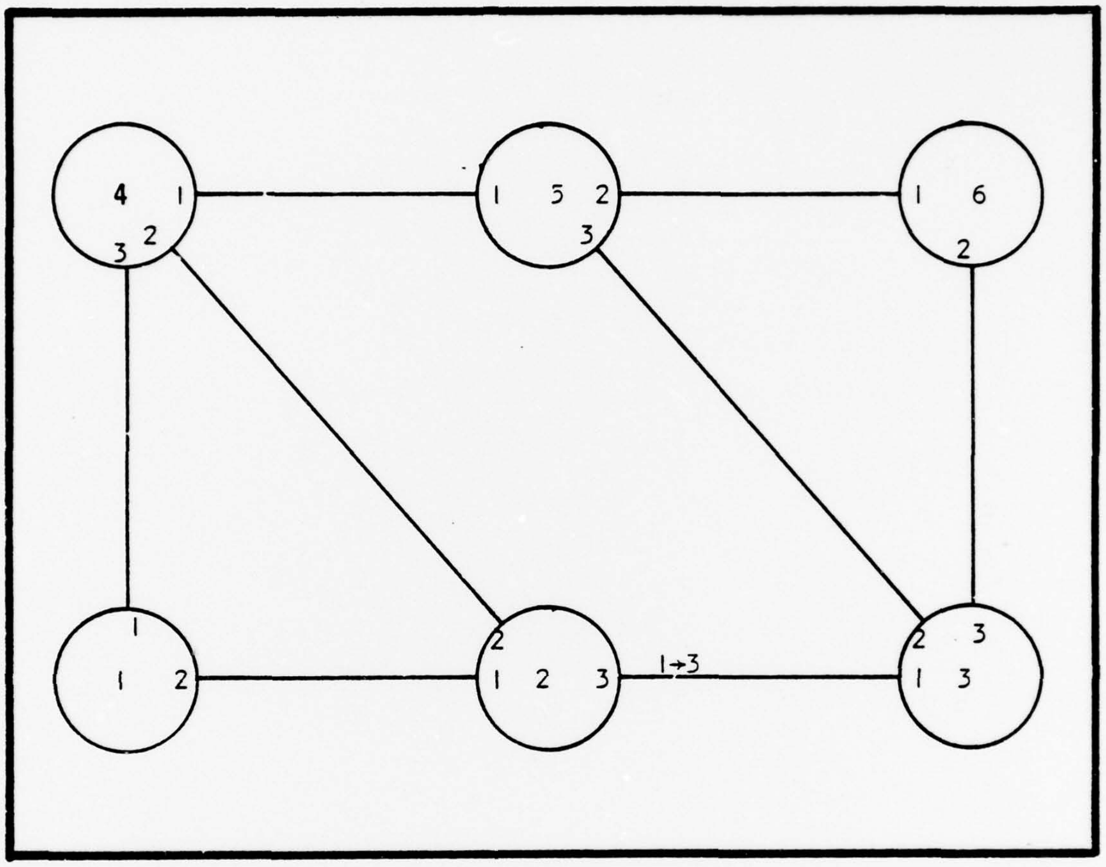


Figure A-1 Reachability Algorithm Example

```
NETWORK ROUTING ALGORITHM

SIMULATION ORDER: 1-TRANSMIT COST MATRIX, 2-RUN
REACHABILITY ALGORITHM.


NETWORK CONFIGURATION:

NODE #    ADJACENT TO    VIA LINE #    COST  BIAS  TIMER
---------------------------------------------------------

  1            4             1          1     0     0
               2             2          1

  2            1             1          1     0     0
               4             2          1
               3             3          1

  3            2             1          1     0     0
               5             2          1
               6             3          1

  4            5             1          1     0     0
               2             2          1
               1             3          1

  5            4             1          1     0     0
               6             2          1
               3             3          1

  6            5             1          1     0     0
               3             2          1

NUMBER OF NODES:   6
VALUE OF MAX:   6

NUMBER OF ITERATIONS:   3
```

```
TIME = 17  ITERATION: 3

NODE 5 TRANSMITTING COST MATRIX.

COST MATRIX (D)

 0          1          2          1          2          3
 1          0          1          1          2          2
 2          1          0          2          1          1
 1          1          2          0          1          2
 2          2          1          1          0          1
 3          2          1          2          1          0

ROUTE DIRECTORY (R)

 0          1          1          3          1          1
 2          0          1          2          1          2
 2          3          0          1          3          2
 1          2          1          0          1          1
 1          2          2          1          0          1
 1          3          3          1          2          0

TIME = 18  ITERATION: 3

NODE 6 TRANSMITTING COST MATRIX.

COST MATRIX (D)

 0          1          2          1          2          3
 1          0          1          1          2          2
 2          1          0          2          1          1
 1          1          2          0          1          2
 2          2          1          1          0          1
 3          2          1          2          1          0

ROUTE DIRECTORY (R)

 0          1          1          3          1          1
 2          0          1          2          1          2
 2          3          0          1          3          2
 1          2          1          0          1          1
 1          2          2          1          0          1
 1          3          3          1          2          0
```

```
THE FOLLOWING CHANGES WERE MADE.

NODE 2 LINE # 3 COST CHANGED TO:  3

NETWORK CONFIGURATION:

NODE #     ADJACENT TO     VIA LINE #      COST   BIAS   TIMER
---------------------------------------------------------------

   1            4               1            1      0      0
                2               2            1

   2            1               1            1      0      0
                4               2            1
                3               3            3

   3            2               1            1      0      0
                5               2            1
                6               3            1

   4            5               1            1      0      0
                2               2            1
                1               3            1

   5            4               1            1      0      0
                6               2            1
                3               3            1

   6            5               1            1      0      0
                3               2            1

NUMBER OF NODES:   6
VALUE OF MAX:   6

NUMBER OF ITERATIONS:   4
```

```
TIME = 1   ITERATION:  1

NODE 1 TRANSMITTING COST MATRIX.

COST MATRIX (D)

 0            1            2            1            2            3
 1            0            1            1            2            2
 2            3            0            2            1            1
 1            1            2            0            1            2
 2            2            1            1            0            1
 3            3            1            2            1            0

ROUTE DIRECTORY (R)

 0            1            1            3            1            1
 2            0            1            2            1            2
 2            1            0            1            3            2
 1            2            1            0            1            1
 1            2            2            1            0            1
 1            2            3            1            2            0

TIME = 2   ITERATION:  1

NODE 2 TRANSMITTING COST MATRIX.

COST MATRIX (D)

 0            1            2            1            2            3
 1            0            1            1            2            2
 3            3            0            2            1            1
 1            1            2            0            1            2
 2            2            1            1            0            1
 3            3            1            2            1            0

ROUTE DIRECTORY (R)

 0            1            1            3            1            1
 2            0            1            2            1            2
 1            1            0            1            3            2
 1            2            1            0            1            1
 1            2            2            1            0            1
 1            2            3            1            2            0
```

```
TIME = 3   ITERATION: 1

NODE 3 TRANSMITTING COST MATRIX.

COST MATRIX (D)

 0         1         2         1         2         3
 1         0         1         1         2         2
 3         3         0         2         1         1
 1         1         2         0         1         2
 2         2         1         1         0         1
 3         3         1         2         1         0

ROUTE DIRECTORY (R)

 0         1         1         3         1         1
 2         0         1         2         1         2
 1         1         0         1         3         2
 1         2         1         0         1         1
 1         2         2         1         0         1
 1         2         3         1         2         0

TIME = 4   ITERATION: 1

NODE 4 TRANSMITTING COST MATRIX.

COST MATRIX (D)

 0         1         2         1         2         3
 1         0         1         1         2         2
 3         3         0         2         1         1
 1         1         2         0         1         2
 2         2         1         1         0         1
 3         3         1         2         1         0

ROUTE DIRECTORY (R)

 0         1         1         3         1         1
 2         0         1         2         1         2
 1         1         0         1         3         2
 1         2         1         0         1         1
 1         2         2         1         0         1
 1         2         3         1         2         0
```

```
TIME = 5   ITERATION:  1

NODE 5 TRANSMITTING COST MATRIX.

COST MATRIX (D)

 0          1          2          1          2          3
 1          0          1          1          2          2
 3          3          0          2          1          1
 1          1          2          0          1          2
 2          2          1          1          0          1
 3          3          1          2          1          0

ROUTE DIRECTORY (R)

 0          1          1          3          1          1
 2          0          1          2          1          2
 1          1          0          1          3          2
 1          2          1          0          1          1
 1          2          2          1          0          1
 1          2          3          1          2          0

TIME = 6   ITERATION:  1

NODE 6 TRANSMITTING COST MATRIX.

COST MATRIX (D)

 0          1          2          1          2          3
 1          0          1          1          2          2
 3          3          0          2          1          1
 1          1          2          0          1          2
 2          2          1          1          0          1
 3          3          1          2          1          0

ROUTE DIRECTORY (R)

 0          1          1          3          1          1
 2          0          1          2          1          2
 1          1          0          1          3          2
 1          2          1          0          1          1
 1          2          2          1          0          1
 1          2          3          1          2          0
```

```
TIME = 7  ITERATION: 2

NODE 1 TRANSMITTING COST MATRIX.

COST MATRIX (D)

 0          1          2          1          2          3
 1          0          1          1          2          2
 3          3          0          2          1          1
 1          1          2          0          1          2
 2          2          1          1          0          1
 3          3          1          2          1          0

ROUTE DIRECTORY (R)

 0          1          1          3          1          1
 2          0          1          2          1          2
 1          2          0          1          3          2
 1          2          1          0          1          1
 1          2          2          1          0          1
 1          2          3          1          2          0

TIME = 8  ITERATION: 2

NODE 2 TRANSMITTING COST MATRIX.

COST MATRIX (D)

 0          1          2          1          2          3
 1          0          1          1          2          2
 3          3          0          2          1          1
 1          1          2          0          1          2
 2          2          1          1          0          1
 3          3          1          2          1          0

ROUTE DIRECTORY (R)

 0          1          1          3          1          1
 2          0          1          2          1          2
 1          2          0          1          3          2
 1          2          1          0          1          1
 1          2          2          1          0          1
 1          2          3          1          2          0
```

```
TIME = 9   ITERATION: 2

NODE 3 TRANSMITTING COST MATRIX.

COST MATRIX (D)

 0          1          2          1          2          3
 1          0          1          1          2          2
 3          3          0          2          1          1
 1          1          2          0          1          2
 2          2          1          1          0          1
 3          3          1          2          1          0

ROUTE DIRECTORY (R)

 0          1          1          3          1          1
 2          0          1          2          1          2
 1          2          0          1          3          2
 1          2          1          0          1          1
 1          2          2          1          0          1
 1          2          3          1          2          0

TIME = 10   ITERATION: 2

NODE 4 TRANSMITTING COST MATRIX.

COST MATRIX (D)

 0          1          2          1          2          3
 1          0          1          1          2          2
 3          3          0          2          1          1
 1          1          2          0          1          2
 2          2          1          1          0          1
 3          3          1          2          1          0

ROUTE DIRECTORY (R)

 0          1          1          3          1          1
 2          0          1          2          1          2
 1          2          0          1          3          2
 1          2          1          0          1          1
 1          2          2          1          0          1
 1          2          3          1          2          0
```

Appendix B

Reachability Algorithm With Hold Down Example

Reachability Algorithm With Hold Down Example

This appendix describes an example using the " hold down " mechanism in the routing algorithm (see Figure 7, page68). The initial network configuration is the same as the initial configuration for Run 2 in Appendix A. The case simulated was the same as in Run 2 of Appendix A (i.e. the cost from node 2 to node 3 increased from one to three). In this run the hold down timer was set to six to allow time for every node in the network to exchange routing information at least once. In this run node 2 uses the previous best path (line 3) to node 3 for the duration of the hold down timer (i.e. from TIME = 1 to TIME = 7 on the simulation output). After the hold down timer expires, node 2 selects line 2 (TIME = 8) as the best path. The significant point is that node 2 did not select the intermediate "best" path (line 1) as node 2 did using only the basic routing algorithm in Appendix A. This means that extraneous information was not transmitted out on line 1 during the search period as it was in Appendix A. Therefore, this example shows how hold down prevents extraneous information from being transmitted through the network during search periods.

```
THE FOLLOWING CHANGES WERE MADE:

NODE 2 LINE # 3 COST CHANGED TO: 3

HOLD DOWN IN EFFECT.


NETWORK CONFIGURATION:

NODE #     ADJACENT TO     VIA LINE #      COST  BIAS  TIMER
-------------------------------------------------------------

   1           4               1             1     0     6
               2               2             1

   2           1               1             1     0     6
               4               2             1
               3               3             3

   3           2               1             1     0     6
               5               2             1
               6               3             1

   4           5               1             1     0     6
               2               2             1
               1               3             1

   5           4               1             1     0     6
               6               2             1
               3               3             1

   6           5               1             1     0     6
               3               2             1

NUMBER OF NODES:   6
VALUE OF MAX:   6

NUMBER OF ITERATIONS:   4
```

```
*** ENTER HOLD DOWN.  NODE:  2 DEST:  3  LINE:  3
*** ENTER HOLD DOWN.  NODE:  2 DEST:  6  LINE:  3

TIME = 1   ITERATION:  1

NODE 1 TRANSMITTING COST MATRIX.

COST MATRIX (D)

 0          1          2          1          2          3
 1          0          1          1          2          2
 2          3          0          2          1          1
 1          1          2          0          1          2
 2          2          1          1          0          1
 3          3          1          2          1          0

ROUTE DIRECTORY (R)

 0          1          1          3          1          1
 2          0          1          2          1          2
 2          3          0          1          3          2
 1          2          1          0          1          1
 1          2          2          1          0          1
 1          3          3          1          2          0
*** ENTER HOLD DOWN.  NODE:  1 DEST:  3  LINE:  2

TIME = 2   ITERATION:  1

NODE 2 TRANSMITTING COST MATRIX.

COST MATRIX (D)

 0          1          2          1          2          3
 1          0          1          1          2          2
 3          3          0          2          1          1
 1          1          2          0          1          2
 2          2          1          1          0          1
 3          4          1          2          1          0

ROUTE DIRECTORY (R)

 0          1          1          3          1          1
 2          0          1          2          1          2
 2          3          0          1          3          2
 1          2          1          0          1          1
 1          2          2          1          0          1
 1          3          3          1          2          0
```

```
TIME = 3   ITERATION: 1

NODE 3 TRANSMITTING COST MATRIX.

COST MATRIX (D)

 0          1          2          1          2          3
 1          0          1          1          2          2
 4          3          0          2          1          1
 1          1          2          0          1          2
 2          2          1          1          0          1
 3          4          1          2          1          0

ROUTE DIRECTORY (R)

 0          1          1          3          1          1
 2          0          1          2          1          2
 2          3          0          1          3          2
 1          2          1          0          1          1
 1          2          2          1          0          1
 1          3          3          1          2          0

TIME = 4   ITERATION: 1

NODE 4 TRANSMITTING COST MATRIX.

COST MATRIX (D)

 0          1          2          1          2          3
 1          0          1          1          2          2
 4          3          0          2          1          1
 1          1          2          0          1          2
 2          2          1          1          0          1
 3          4          1          2          1          0

ROUTE DIRECTORY (R)

 0          1          1          3          1          1
 2          0          1          2          1          2
 2          3          0          1          3          2
 1          2          1          0          1          1
 1          2          2          1          0          1
 1          3          3          1          2          0
```

```
TIME = 5   ITERATION: 1

NODE 5 TRANSMITTING COST MATRIX.

COST MATRIX (D)

 0          1          2          1          2          3
 1          0          1          1          2          2
 4          3          0          2          1          1
 1          1          2          0          1          2
 2          2          1          1          0          1
 3          4          1          2          1          0

ROUTE DIRECTORY (R)

 0          1          1          3          1          1
 2          0          1          2          1          2
 2          3          0          1          3          2
 1          2          1          0          1          1
 1          2          2          1          0          1
 1          3          3          1          2          0

TIME = 6   ITERATION: 1

NODE 6 TRANSMITTING COST MATRIX.

COST MATRIX (D)

 0          1          2          1          2          3
 1          0          1          1          2          2
 4          3          0          2          1          1
 1          1          2          0          1          2
 2          2          1          1          0          1
 3          4          1          2          1          0

ROUTE DIRECTORY (R)

 0          1          1          3          1          1
 2          0          1          2          1          2
 2          3          0          1          3          2
 1          2          1          0          1          1
 1          2          2          1          0          1
 1          3          3          1          2          0
```

```
### EXIT HOLD DOWN.  NODE:  2 DEST.  3
### EXIT HOLD DOWN.  NODE.  2 DEST.  6

TIME =  7   ITERATION:  2

NODE  1  TRANSMITTING COST MATRIX.

COST MATRIX (D)

 0          1          2          1          2          3
 1          0          1          1          2          2
 4          3          0          2          1          1
 1          1          2          0          1          2
 2          2          1          1          0          1
 3          4          1          2          1          0

ROUTE DIRECTORY (R)

 0          1          1          3          1          1
 2          0          1          2          1          2
 2          3          0          1          3          2
 1          2          1          0          1          1
 1          2          2          1          0          1
 1          3          3          1          2          0
### EXIT HOLD DOWN.  NODE:  1 DEST:  3

TIME =  8   ITERATION:  2

NODE  2  TRANSMITTING COST MATRIX.

COST MATRIX (D)

 0          1          2          1          2          3
 1          0          1          1          2          2
 4          3          0          2          1          1
 1          1          2          0          1          2
 2          2          1          1          0          1
 3          3          1          2          1          0

ROUTE DIRECTORY (R)

 0          1          1          3          1          1
 2          0          1          2          1          2
 2          2          0          1          3          2
 1          2          1          0          1          1
 1          2          2          1          0          1
 1          2          3          1          2          0
```

```
TIME = 9  ITERATION: 2

NODE 3 TRANSMITTING COST MATRIX.

COST MATRIX (D)

 0          1          2          1          2          3
 1          0          1          1          2          2
 3          3          0          2          1          1
 1          1          2          0          1          2
 2          2          1          1          0          1
 3          3          1          2          1          0

ROUTE DIRECTORY (R)

 0          1          1          3          1          1
 2          0          1          2          1          2
 1          2          0          1          3          2
 1          2          1          0          1          1
 1          2          2          1          0          1
 1          2          3          1          2          0

TIME = 10  ITERATION: 2

NODE 4 TRANSMITTING COST MATRIX.

COST MATRIX (D)

 0          1          2          1          2          3
 1          0          1          1          2          2
 3          3          0          2          1          1
 1          1          2          0          1          2
 2          2          1          1          0          1
 3          3          1          2          1          0

ROUTE DIRECTORY (R)

 0          1          1          3          1          1
 2          0          1          2          1          2
 1          2          0          1          3          2
 1          2          1          0          1          1
 1          2          2          1          0          1
 1          2          3          1          2          0
```

Appendix C

Routing Simulation Program Listing

```
10 LINE 132
20 INPUT "DESTINATION(0=CRT,3=PRINTER) ",D
30 INPUT"LONG OR SHORT PRINTOUT (1 OR 0) ? ",F2
40 PRINT#D,CHR$(26)
50 PRINT#D,CHR$(12),"NETWORK ROUTING ALGORITHM"\PRINT#D,
60 PRINT#D,"SIMULATION ORDER: 1-TRANSMIT COST MATRIX, 2-RUN "
70 PRINT#D,"REACHABILITY ALGORITHM."\PRINT#D,
80 DIM C1(7,7)\REM NODE VS LINE=DEST NODE
90 DIM C2(7,7)\REM NODE VS SOURCE=IN LINE
100 DIM C3(7)\REM # LINES/NODE
110 DIM U(7,7,7),D(7,7),R(7,7),A(7,7)
120 DIM H(7,7),D1(7),R1(7),H1(7),B(7)
130 M=6\N=6
140 FORI=1TON\READC3(I)\NEXT
150 FORI=1TON\FORJ=1TON-1\READC1(I,J)\NEXT\NEXT
160 FORI=1TON\FORJ=1TON\READC2(I,J)\NEXT\NEXT
170 FORI=1TON-1\FORJ=1TON\READA(I,J)\NEXT\NEXT
180 FORI=1TON\READB(I)\NEXTI
190 FORI=1TON\READH1(I)\NEXT
200 REM INITIALIZE DATA
210 FORI=1TON\FORJ=1TON\D(I,J)=M\FORK=1TON\U(I,J,K)=M
215 NEXTK\NEXTJ\NEXTI
220 INPUT"COST CHANGES (Y OR N)",C$\IFC$<>"Y"THEN320
230 PRINT#D,\PRINT#D,\PRINT#D,"THE FOLLOWING CHANGES WERE MADE:"
240 PRINT#D,
250 INPUT"ENTER: NODE,LINE,COST (< 0 TO STOP) ",I,J,C
260 IFI<1ORI>NORJ<1ORJ>N-1THEN250
270 IFC<0THEN320\REM STOP CHANGES
280 A(J,I)=C
290 PRINT#D,"NODE",I," LINE #",J," COST CHANGED TO:",A(J,I)
300 GOTO 250
310 PRINT#D,
320 INPUT"HOLD DOWN DESIRED (Y OR N) ",C$
321 IF C$="Y" THEN F1=1 ELSE F1=0
330 IFF1=0THEN GOTO350
340 PRINT#D,\PRINT#D,"HOLD DOWN IN EFFECT."\PRINT#D,
350 INPUT"NETWORK CONFIGURATION PRINTOUT(Y OR N) ",C$
351 IF C$= "N" THEN 500
360 D9=D
370 INPUT"CONFIG PRINTOUT DEST (0=CRT,1=AUX CRT,3=PRINTER,4=AUX CRT) ",D
380 PRINT#D,
390 PRINT#D,\PRINT#D,"NETWORK CONFIGURATION:"\PRINT#D,\PRINT#D,
400 PRINT#D,"NODE #",TAB(10),"ADJACENT TO",TAB(25),"VIA LINE #",TAB(40),
410 PRINT#D,"COST",TAB(46),"BIAS",TAB(52),"TIMER"
420 FORI=1TO57\PRINT#D,"-",\NEXT\PRINT#D,\PRINT#D,
430 FORI=1TON\PRINT#D,TAB(2),I,
440   FORJ=1TOC3(I)
450   PRINT#D,TAB(15),C1(I,J),TAB(28),J,TAB(41),A(J,I),
460    IFJ>1THEN470ELSEPRINT#D,TAB(47),B(I),TAB(53),H1(I),
470   PRINT#D,\NEXT J\PRINT#D,
480 NEXT I
490 D=D9
500 PRINT#D,"NUMBER OF NODES: ",N
510 PRINT#D,"VALUE OF MAX: ",M
520 PRINT#D,
530 INPUT"NUMBER OF SIMULATION ITERATIONS",N1
540 PRINT#D,"NUMBER OF ITERATIONS: ",N1
550 INPUT"ITERATIONS PER PAGE ",P1\IFP1<=0THEN550
560 PRINT#D,CHR$(12)
570 DATA 2,3,3,3,3,2
580 REM C1
590 DATA 4,2,0,0,0
600 DATA 1,4,3,0,0
610 DATA 2,5,6,0,0
620 DATA 5,2,1,0,0
630 DATA 4,6,3,0,0
640 DATA 5,3,0,0,0
650 REM C2
660 DATA 0,2,0,1,0,0
670 DATA 1,0,3,2,0,0
680 DATA 0,1,0,0,2,3
690 DATA 3,2,0,0,1,0
700 DATA 0,0,3,1,0,2
710 DATA 0,0,2,0,1,0
720 REM A
730 DATA 1,1,1,1,1,1
740 DATA 1,1,1,1,1,1
750 DATA 0,1,1,1,1,0
760 DATA 0,0,0,0,0,0
770 DATA 0,0,0,0,0,0
780 REM B
790 DATA 0,0,0,0,0,0
800 REM H1(HOLD INIT VAL)
810 DATA 0,0,0,0,0,0
820 T9=0\REM INITIALIZE SIMULATED TIME.
830 FORI8=1TON1\REM MAIN LOOP COUNTER.I.E. # OF COMPLETE ITERATIONS
840             REM THROUGH SIMULATION.
850 FORI7=1TON\REM I7=CURRENT NODE TRANSMITTING COST MATRIX
860 T=FNA(I7)
870 T9=T9+1\REM T9 IS SIMULATED TIME IN TICKS.
880 FORI9=1TON\REM I9=CURRENT IMP
890   FORI=1TON\REM I=DEST IMP #
900 IF I=I9THEN1120\REM DO NOT PROCESS SELF
910   IF H(I,I9) > 0 THEN 1070\ REM IN HOLD DOWN FOR I?
920   D1(I9)=M\REM SET TEMP COST TO MAX
930   R1(I9)=R(I,I9)\REM SET TEMP LINE
940      FORJ=1TOC3(I9)\REM J=LINE # FOR IMP I9
950      IF U(I,J,I9)+A(J,I9)+B(I9)>=D1(I9)THEN 980
960      D1(I9)=U(I,J,I9)+A(J,I9)+B(I9)\REM UPDATE TEMP COST TO I
970      R1(I9)=J\REM UPDATE NEW TEMP LINE TO I
980      T=U(I,J,I9)+A(J,I9)+B(I9)
990      IF J<> R(I,I9) OR T<= D(I,I9) THEN 1020
1000      H(I,I9)=H1(I9)\REM SET HOLD DOWN TIMER
1009      IF F1=0 THEN 1020
1010      PRINT#D,"*** ENTER HOLD DOWN. NODE:",I9," DEST:",
1012      PRINT#D,I,"  LINE:",J
1020      NEXT J
1030    D(I,I9)=D1(I9)\REM SET NEW COST TO )
```

```
1040      IF H(I,I9)=0THEN R(I,I9)=R1(I9)\REM IF IN HOLD DOWN DO NOT
1050                                             REM UPDATE ROUTE.
1060     GOTO 1120
1070     D(I,I9)=U(I,R(I,I9),I9)+A(R(I,I9),I9)+B(I9)\REM IN HOLD DOWN.
1080                        REM SO UPDATE COST TO CURRENT VALUE OF PREVIOU
1090                               REM BEST ROUTE.
1100     H(I,I9)=H(I,I9)-1\REM DECREMENT HOLD DOWN COUNTER
1109     IF H(I,I9)<>0 OR F1<>1 THEN  1120
1110     PRINT#D,"### EXIT HOLD DOWN. NODE:",I9," DEST:",I
1120    NEXT I
1130    D(I9,I9)=0
1140    R(I9,I9)=0\REM SET SELF REACHABILITY TO 0
1150     H(I9,I9)=0
1160 NEXT I9
1170 PRINT#D,\PRINT#D,"TIME =",T9,"  ITERATION:",I8\PRINT#D,
1180 PRINT#D,"NODE",I7," TRANSMITTING COST MATRIX."
1190 IF F2=0THEN1280
1200 PRINT#D,"NEIGHBORS' COST MATRIX (NC)"\PRINT#D,
1210 FORI=1TON\PRINT#D, " NODE",I,TAB(13*I+1),\NEXTI\PRINT#D,\PRINT#D,
1220 FORI=1TON\FORK=1TON\FORJ=1TON
1230 PRINT#D,%3I, U(I,J,K),
1240 NEXT J
1250 PRINT#D, TAB(13*K),
1260 NEXT K
1270 PRINT#D,\NEXT I
1280 PRINT#D,
1290 PRINT#D,"COST MATRIX (D)"\PRINT#D,
1300 FORI=1TON\FORJ=1TON
1310 PRINT#D, D(I,J),TAB(12*J),\NEXT J
1320 PRINT#D,
1330 NEXTI
1340 PRINT#D,\PRINT#D,"ROUTE DIRECTORY (R)"\PRINT#D,
1350 FORI=1TON\FORJ=1TON\PRINT#D, R(I,J),TAB(12*J),\NEXT J
1360 PRINT#D,\NEXT I
1370 IFI7-INT(I7/P1)*P1=0THENPRINT#D,CHR$(12)
1380 NEXT I7
1390 NEXT I8\REM ANOTHER ITERATION.
1400 INPUT"CONTINUE (Y OR N) ",C$\IFC$="Y"THEN 1420
1410 STOP
1420 INPUT"DESTINATION (0=CRT,1=AUX CRT,3=PRINTER,4=AUX CRT) ? ",D
1430 GOTO 220
1440 DEF FNA(T)
1450 REM PASSES THE D ARRAY OUT OF ALL LINES FOR IMP T/
1460 FOR T1=1TO C3(T)\REM T1=OUTPUT LINE #
1470 L=C2(C1(T,T1),T)\REM RECEIVING IMP'S LINE #
1480 P=C1(T,T1)\REM RECEIVING IMP #
1490    FOR T2=1TON\REM T2=DEST IMP
1500    U(T2,L,P)=D(T2,T)\REM TRANSFER D ARRAY FROM IMP T TO P
1510    NEXT T2
1520 NEXT T1
1530 RETURN T
1540 FNEND
```

Appendix D

CIN3 Program Listing

```
               ORG     X'FC00'
               EI
               NOP
               NOP
               NOP
               NOP
;                                 DATA MODE.
PORT0          IN      020        CRT INPUT
               ANI     01         CRT READY?
               JZ      PRT00I     ;CHECK OTHER PORTS IF NOT READY
               IN      021
               ANI     0177       MASK BITS 0-7
               CPI     024        ;CONTROL T ?
               CZ      TELE1      YES.
               NOP
               RET
TELE1          PUSH    B
               PUSH    H          ;SAVE HL
               LHLD    ATMSG      ;LOAD TMSG POINTER.
               MVI     A,00       ;SET OUT FLAG TO CRT
               CALL    MSG
               POP     H          ;RESTORE HL
TELE           IN      020        CRT STATUS
               ANI     01         READY?
               JZ      TPORT1     NO,CHECK PORT1 FOR INPUT
               IN      021        ELSE INPUT FROM PORT 0
               ANI     0177
               CPI     024        CONTROL T INPUT?
               JZ      CONTG      YES,EXIT TELE MODE.
               MOV     B,A
               MVI     A,00       SET CRT OUT FLAG.
               CALL    COUT       ECHO CHARCTER.
               MVI     A,02       SET CYBER OUT FLAG
               CALL    COUT       OUTPUT TO CYBER.
TPORT1         IN      022        INPUT PORT 1 STATUS.
               ANI     01
               JZ      TELE       NOT READY
               IN      023
               MOV     B,A
               MVI     A,00       SET CRT OUT FLAG
               CALL    COUT       OUTPUT TO CRT.
               JMP     TELE       LOOK FOR NEW INPUT.
CONTG          MVI     A,0        SET MODE TO 0
               STA     MODE
               PUSH    H
               LHLD    ADMSG      ;LOAD DMSG POINTER.
               MVI     A,00       ;SET OUT FLAG TO CRT
               CALL    MSG
               POP     H
               POP     B
```

```
                LDA     CR      PUT CR IN A FOR DOS
                RET
                NOP
                NOP
        CYIN    EI
                PUSH    PSW     ;SAVE STATUS
                PUSH    D
                PUSH    H
        CYRDY   IN      020
                ANI     01
                JZ      CYBIN   ;JUMP IF CRT NOT READY
                IN      021
                ANI     0177
                CPI     004     ;CONTROL D?
                JZ      CONTD   ;YES
                CPI     020     ;CONTROL P?
                CZ      CONTP
                CPI     021     ;CONTROL Q?
                CZ      CONTQ
                CPI     023     ;CONTROL S
                CZ      CONTS
        CYBIN   IN      022     ;PORT 1 STATUS
                ANI     01
                JZ      CYRDY   ;JUMP IF NOT READY
                IN      023     INPUT DATA FROM CYBER.
                ANI     0177    MASK 7 LSB
                CPI     0       ;CHECK FOR NULL
                JZ      CYDONE
                PUSH    PSW     ;SAVE DATA READ
                MOV     B,A
                LDA     PRNTTY
                CPI     0200    ;PRINT ?
                CNZ     COUT
                LDA     EDIT
                CPI     01      ;IN EDIT MODE ?
                JZ      LEDIT   ;JUMP IF IN EDIT.
                LHLD    TACOM   ;GET CURRENT POINTER IN COM
                JMP     CHECK
        LEDIT   LHLD    TAEDT   GET EDT CURRENT POINTER
        ;                       HL NOW CONTAINS POINTER
        ;                       TO CURRENT DELIMITER
        ;                       CHARACTR.
        CHECK   POP     PSW
                CMP     M
                PUSH    PSW     ;RESTORE PSW TO KEEP STACK
        ;                       CORRECT. A STILL CONTAINS
        ;                       DATA.
                JZ      MATCH   ;JUMP IF MATCH
                MOV     A,M     LOAD EOTC.
                LHLD    AEOTC   ;GET POINTER TO EOTC
                CMP     M
        JZ      LEOTC    JUMP IF EOTC FOUND.
        LHLD    AEDT     ;RESET POINTERS TO
        SHLD    TAEDT    ;BEGINNING OF
        LHLD    ACOM     ;DELIMITERS.
        SHLD    TACOM
        POP     PSW
        CMP     M
        PUSH    PSW
        JZ      MATCH
        NOP
        NOP
        NOP
        NOP
ECYIN   LHLD    CYCNT    ;LOAD CURRENT CFILE COUNT
        DCX     H
;*** CHECK FOR OVERFLOW
        MOV     A,H
        CPI     00
        JNZ     CYSTOR   ;NOT 0 SO CONTINUE
        MOV     A,L      ;CHECK L REG
        CPI     00
        JNZ     CYSTOR   ;JUMP IN NOT ZERO
        MVI     A,01     ;BUFFER FULL
        STA     CYFULL
        POP     PSW      ;KEEP STACK STRAIGHT
        JMP     CYDONE
CYSTOR  SHLD    CYCNT    STORE CURRENT COUNT.
        LHLD    CYFW     ;GET FIRST WORD POINTER
        XCHG
        LHLD    CYCUR    ;POINTER TO LAST WORD STORED
        INX     H
        SHLD    CYCUR    ;CYCUR=*+1
        DAD     D        RELATIVE ADDRESS IN CFILE
        POP     PSW      ;RETRIEVE DATA
        MOV     M,A      ;STORE DATA
        JMP     CYRDY
CYDONE  POP     H        ;RESTORE STACK
        POP     D
        POP     PSW
        RET
MATCH   LHLD    TACOM
        INX     H        ;TACOM=*+1
        SHLD    TACOM
        LHLD    TAEDT
        INX     H
        SHLD    TAEDT    ;TAEDT=*+1
        JMP     ECYIN
LEOTC   MVI     A,01
        STA     EOT      ;SET EOT=1
        LHLD    AEDT     ;RESET POINTERS
        SHLD    TAEDT
        LHLD    ACOM
        SHLD    TACOM
        LHLD    CYCNTI
        SHLD    CYCNT
        POP     PSW       ;RETRIEVE DATA
        POP     H
        POP     D
        POP     PSW
        RET
CONTD   LHLD    AABMSG    ;ABORT MSG POINTER
        MVI     A,00      ;SET CRT OUT FLAG
        CALL    MSG
        PUSH    PSW       ;KEEP STACK CORRECT.
        JMP     LEOTC     ;RESET POINTERS.
CONTP   PUSH    PSW
        LDA     PRNTTY    ;GET PRINT MASK
        ORI     010       ;SET PRINTER BIT
        STA     PRNTTY
        MVI     A,01
        STA     PRNTF     ;SET PRINTF FLAG
        POP     PSW
        RET
CONTQ   PUSH    PSW
        LDA     PRNTTY
        MVI     A,0200
        STA     PRNTTY    ;SET CRT IN MASK
        STA     PRNTF     ;SET PRINT FLAG
        POP     PSW
        RET
CONTS   PUSH    PSW
        MVI     A,00
        STA     PRNTTY    ;RESET PRINT FLAG
        POP     PSW
        RET
COUT    EI
        PUSH    PSW       ;SAVE A REG
        ANI     07        ;MASK 3 LSB
        CPI     0
        JNZ     CYOUT     ;CHECK OUT TO CYBER
        DI                ;DI IF OUT TO DEVICE 0
        NOP
        NOP
        NOP
        NOP
CRTRDY  IN      020
        ANI     02
        JZ      CRTRDY    NOT READY.
        MOV     A,B
        OUT     021
```

```
                ORG     X'FB00'
                EI
                STA     INMASK   ;SAVE INPUT MASK
       PRT00I   LDA     INMASK
                CPI     0
                JNZ     PRT24I   ;CHECK PORT 024 IF NOT PORT 0
                JMP     PORT0    ;ELSE JUMP TO PORT 0
       PRT24O   POP     PSW      ;PORT 024
                PUSH    PSW
                CPI     01       ;CHECK PORT 024 DEVICE FLAG
                JNZ     PRT26O   ;NOT SET SO JUMP
       PRT24Y   IN      024
                ANI     02
                NOP
                NOP
                NOP
                NOP
                JZ      PRT24Y   ;NOT RDY
                MOV     A,B
                OUT     025
       PRT26O   POP     PSW
                PUSH    PSW
                CPI     04
                JNZ     NOOUT
       PRT26Y   IN      026
                ANI     02
                NOP
                NOP
                NOP
                NOP
                JZ      PRT26Y
                MOV     A,B
                OUT     027
                JMP     NOOUT
       PRT24I   LDA     INMASK   ;LOAD INPUT MASK
                CPI     01       ;INMASK=1 ?
                JNZ     PRT26I   ;NO-CHECK PORT 026
       P24RDY   IN      024      ;INPUT STATUS
                ANI     01
                NOP
                NOP
                NOP
                NOP
                JZ      PORT0    ;THIS ALLOWS PORT 0 TO ALWAYS
       ;                          RECEIVE INPUT EVEN IF PORT 024
       ;                          IS NOT READY.
                IN      025
                ANI     0177
                NOP
                NOP
                NOP
                RET
```

```
CYOUT   POP     PSW
        PUSH    PSW
        CPI     02
        JNZ     PRNT
CYBRDY  IN      022
        ANI     02      CYBER READY?
        JZ      CYBRDY  NO.
        MOV     A,B
        OUT     023     OUT TO CYBER.
        CALL    TIRST
PRNT    POP     PSW
        PUSH    PSW
        CPI     03
        JZ      PRNTRY  ;JUMP IF OUT TO PRINTER
        ANI     010
        JZ      PRT240  ;CHECK PORT 024 OUT
PRNTRY  IN      042     PRINTER STATUS
        ANI     0200
        JZ      PRNTRY  NOT READY.
        IN      043     RESEET CONTROL BIT 7
        MOV     A,B
        OUT     043
        JMP     NOOUT
MSS     STA     PRNTTY
MSGLP   LDA     PRNTTY
;                       ;POINTED TO BY HL
        MOV     B,M     ;MOVE CHARACTER TO BE PRINTED
        CALL    COUT
        INX     H       ;INCREMENT POINTER.
        MOV     A,M
        CPI     0       ;EOM CHARACTER ?
        RZ              ;YES, SO RETURN.
        JMP     MSGLP
FLOOK   PUSH    H       ;LOOK FOR FILE ON MINI-FLOPPY
        PUSH    PSW
        MVI     A,01    ;A MUST BE=1 FOR DLOOK
        STA     FPRESF  ;ASSUME FILE FOUND
        LHLD    AFNAME
        DI
        CALL    X'201C' DOSCYB1 DLOOK ROUTINE
        EI
        JNC     FLDONE  ;FILE FOUND
        MVI     A,0
        STA     FPRESF  ;NOT FOUND
FLDONE  POP     PSW
        POP     H
        RET
;
;
;
```

```
021715  PRT26I  LDA     INMASK
021720          CPI     04      ;INMASK=4 ?
021725          JNZ     NEXTIN  ;CHECK NEXT INPUT DEVICE
021730  P26RDY  IN      026     ;PORT 026 INPUT ROUTINE
021735          ANI     01
021743          NOP
021745          NOP
021750          NOP
021753          NOP
021755          JZ      PORT0   ;SAME AS PORT 024
021760          IN      027
021765          ANI     0177
021770          NOP
021775          NOP
021780          NOP
021785          RET
```

```
         ORG    0176732
AABMSG   DW     ABMSG
ATMSG    DW     TMSG
ADMSG    DW     DMSG     ;ADDRESS OF DMSG
         NOP
         NOP
         NOP
         NOP
ACOM     DW     COM      ;POINTER TO START OF COM
AEDT     DW     EDT      ;POINTER TO START OF EDT
AEOTC    DW     EOTC     POINTER TO EOTC.
AFNAME   DW     FNAME
         NOP
         NOP
         NOP
TIRST    PUSH   H
         LXI    H,TICNT
         MVI    A,00
         MOV    M,A
         INX    H
         MOV    M,A      SECONDS
         INX    H
         MOV    M,A      ;MINUTES
         INX    H
         MOV    M,A      ;HOURS
         POP    H
         RET
         NOP
         NOP
         ORG    X'2915'
NEXTIN   JMP    PORT0
         ORG    X'29C0'
NOOUT    POP    PSW      ;RESTORE STACK
         MOV    A,B      ;DATA MUST BE RETURNED IN A.
         RET
         ORG    000
LEV0     JMP    LEV0
         ORG    010
LEV1     PUSH   D
         PUSH   H
         PUSH   PSW
         JMP    CYIN
         ORG    020
LEV2     JMP    LEV2
         ORG    030
LEV3     JMP    LEV3
         ORG    040
LEV4     JMP    LEV4
         ORG    050
LEV5     JMP    LEV5
         ORG    060
LEV6     JMP    LEV6
```

```
              ORG 0100         ;DATA
     MODE     DS       1
     CR       DB       0       ;NULL
     EOT      DS       1       ;END OF TEXT FLAG
     CRG      DS       1       ;MAX VALUE FOR I
     EDIT     DS       1
     COM      DB       012
              DC       "COMMAND- "
     I        DS       1       ;COM AND EDT INDEX
     EDT      DB       012
              DC       ". "
     EDTC     DB       0240
     CYCUR    DS       2        ;(L,H)  CFILE CURRENT POINTER
     CYFW     DB       000,001 ;(L,H)  CFILE FIRST WORD POINTER
     CYCNTI   DB       000,020 ;(L,H)  INITIAL MAX COUNT
     CYCNT    DS       2        ;(L,H)   CFILE CURRENT COUNT
     CYFULL   DS       1        ;CFILE FULL FLAG.
     FNAME    DS       9        ;FILE NAME
     FPRESF   DS       1        ;=1 FILE FOUND,=0 NOT FOUND
     NMB      DS       5        ;TIME COUNTERS
     CURLEV   DB       0        ;INTERUPT MASK
     TMSG     DB       012,015,"TELE MODE",07,07,015,012,0
     DMSG     DB       012,015,"DATA MODE",07,015,012,0
     ABMSG    DB       012,015,"ABORT CYIN",07,07,015,012,0
     PRNTTY   DB       0        ;PRINT MASK
     PRNTF    DB       0        ;PRINT FLAG
     TACOM    DW       COM      ;TEMP POINTER TO COM
     TAEDT    DW       EDT      ;TEMP POINTER TO EDT
     TICNT    DS       4        ;CURRENT DURATION COUNTER
     TISET    DB       0,0,5,0 ;INTERUPT ALARM INTERVAL(.01,SEC,MIN,HF
     TIFLG    DB       01       ;TIME DURATION FLAG
     TITRP1   JMP      X'2900' ;TIFLG=001
     TITRP2   DS       3        ;TIFLG=002
     TITRP3   DS       3        ;TIFLG=004
     TITRP4   DS       3        ;TIFLG=010
     TITRP5   DS       3        ;TIFLG=020
     TITRP6   DS       3        ;TIFLG=040
     TITRP7   DS       3        ;TIFLG=100
     TITRP8   DS       3        ;TIFLG=200
     INMASK   DB       0        ;INPUT DEVICE MASK
              END
```

Appendix E

CYBER Program Listing

```
100 REM ******************* CYBER **********************
200 REM ***************** 18 SEP 78 *******************
300 REM
400 DIM C$(72),A(2),T4(4),T$(500)
500 M1=256*40-1\REM MAX <DEST> SIZE
600 M=64\REM MODE
700 E=66\REM EOT
800 E1=68\REM EDIT
900 C=64613\REM CYIN(176145)
1000 C1=84\REM CYCUR
1100 C2=86\REM CYFW
1200 C3=90\REM CYCNT
1300 C4=92\REM CYFULL
1400 C5=88\REM CYCNTI
1500 F=64960\REM FLOOK(176700)
1600 C6=102\REM FPRESF
1700 C7=93\REM FNAME
1800 T1=103\REM NMB
1900 T3=65024\REM TINIT (177000)
1910 T5=65221\REM INTERUPT ENABLE(177305)
1920 T6=65223\REM      "      DISABLE(177307)
1930 T7=167\REM TISET
1940 T8=171\REM TIFLG
1950 T9=163\REM TICNT
2000 FILL M,0\FILL E,0\FILL C4,0\FILL E1,0\FILL T8,1
2100 T=CALL(T3)\REM INITIALIZE VI BOARD/RTC
2200 D=0\PRINT\INPUT"REQUEST- ",C$
2300 R=FNR(C$)
2400 ON R GOTO 2500,2600,2700,2800,2900,3000,3100,3200,3300
2500 E2=1\T=FNE(E2)\GOTO 2200
2600 F1=0\GOSUB 5200\REM "CY"\GOTO 2200
2700 GOTO 2200\REM BLANK LINE
2800 GOSUB 18100\GOTO2200\REM "TR"
2900 GOSUB 24100\GOTO2200\REM "SA"
3000 GOSUB29000\GOTO2200\REM LF
3100 GOSUB 6900\GOTO2200\REM "LC"
3200 GOSUB 9600\GOTO2200\REM "TI"
3300 GOSUB14600\PRINT\PRINT"END OF PROGRAM: ",
3310 GOSUB10900\END
3400 REM ***********************************************
3500 DEF FNE(E2)
3600 PRINT #D,
3700 ON E2 GOTO 3800,4000,4200,4400,4500,4600,4700,4800,4900
3800 PRINT #D,"ILLEGAL REQUEST- ",C$(P1,LEN(C$))
3900 GOTO5000
4000 PRINT#D,"ARGUMENT ERROR- ",C$(P1,LEN(C$))
4100 GOTO5000
4200 PRINT#D,"ARG OUT OF CFILE- ",C$(P1,LEN(C$))
4300 GOTO5000
4400 PRINT#D,"NUMERIC ARG EXPECTED- ",C$(P1,LEN(C$))\GOTO5000
```

```
4500 PRINT#D,"<SOURCE> DOES NOT EXIST- ",S$\GOTO5000
4600 PRINT#D,"EMBEDDED BLANK IN ARG- ",C$(P2,P1)\GOTO5000
4700 PRINT#D,"MISSING ARGUMENT- ",C$(1,LEN(C$))\GOTO5000
4800 PRINT#D,"<DEST> DOES NOT EXIST- ",D$\GOTO5000
4900 PRINT#D,"<DEST> FILE FULL- ",D$\GOTO5000
5000 PRINT #D,\RETURN E2
5100 FNEND
5200 REM ************************************************
5300 REM... "CY" REQUEST
5400 FILL E,0\REM SET EOT TO 0
5410 REM SET POINTER TO BEGINNING OF CFILE
5500 FILL C1,0\FILL C1+1,0
5600 T=CALL(C)\REM CALL CYIN
5700 IF EXAM(E)=0THEN 6300
5800 IFF1=1THENRETURN
5900 GOSUB14500\REM..GET TIME
6000 PRINT
6100 PRINT#D,"CYBER MESSAGE RECEIVED. ",T4(3),
6110 PRINT#D,":",T4(2),":",T4(1)
6200 PRINT\RETURN
6300 REM OVERFLOW
6400 PRINT#D,"BUFFER FULL ",FNA(C5)," BYTES."
6500 PRINT#D,\FILL C4,0\REM RESET CYFULL
6600 FILL C3,EXAM(C5)\FILL C3+1,EXAM(C5+1)\REM COUNT
6700 RETURN
6800 REM...END CY
6900 REM ************************************************
7000 REM LC,<START>,<STOP>(,<DEVICE>)
7100 IF P1+2>LEN(C$)THEN 26000
7200 P1=P1+2
7300 FORI7=0 TO 2
7400 IFP1<=LEN(C$)THEN7500\P1=4\EXIT 8500
7500 IF C$(P1,P1)<>","THEN26000
7600 P1=P1+1\V7=FNN(P1)\REM..NUMBER?
7700 IFI7<>2THEN7900
7800 IFV7=0THEN EXIT 8500\REM NO DEVICE CODE
7900 IFV7=0THENEXIT26200\REM..NOT A NUMBER
8000 A(I7)=FNV(P1)\REM..GET VALUE OF NUMBER
8100 IFI7<>2THEN8200\D=A(I7)\EXIT 8500
8200 IF A(I7)>=0THEN8300ELSEEXIT26100
8300 IF A(I7)<=FNA(C5)THEN8400ELSEEXIT26100
8400 NEXT
8500 IF A(1)<A(0)THEN26000\REM STOP < START
8600 A4=FNA(C2)+1\REM ADDRESSOF BEGINNING OF CFILE
8700 T=FNP(A(0),A(1),A4,70)
8800 PRINT"COLUMN PRINTOUT DESIRED (YES OR NO) ? ",
8810 INPUT" ",C$
8900 IF C$<>"YES"THEN RETURN
9000 PRINT#D,\PRINT#D," LOC ",TAB(10),"DEC",TAB(18),"ASCII"
9100 FOR I=1TO21\PRINT#D,"-",\NEXT\PRINT#D,\PRINT#D,
9200 FOR I=A(0) TO A(1)
9300 PRINT #D,%5I,I,TAB(10),%3I,EXAM(I+A4),TAB(18),
9310 PRINT#D,CHR$(EXAM(A4+I))
9400 NEXT
```

```
9500 RETURN
9600 REM ***********************************************
9700 REM TI(,<HOURS>,<MINS>,<SECS>(,DUR)) ROUTINE
9800 IF P1+2>LEN(C$)THEN10800\REM   DISPLAY
9900 P1=P1+2
10000 FORI8=0TO2
10100 IFP1<=LEN(C$)THEN10200\P1=4\EXIT 26000
10200 IFC$(P1,P1)<>","THENEXIT26000\P1=P1+1
10210 IFFNN(P1)=0THENEXIT26200
10300 T4(3-I8)=FNV(P1)\NEXT\P2=P1\P1=4
10400 IFT4(3)>23THEN26000ELSEIFT4(2)>59THEN26000
10410 IFT4(1)>59THEN26000
10500 IFP2>LEN(C$)THEN10600
10510 IFC$(P2,P2+3)<>",DUR"THEN26000
10520 FORI=0TO3\FILL T7+I,T4(I)\FILL T9+I,0\NEXT\RETURN
10600 FORI=0TO3\FILL  T1+I,T4(I)\NEXT
10700 RETURN
10800 GOSUB 14500
10900 PRINT#D,"TIME:  ",T4(3),":",T4(2),":",T4(1)\PRINT#D,
11000 RETURN
11100 REM ***********************************************
11200 DEF FNR(C$)\REM..SCANS C$ FOR COMMAND.
11300 REM  RETURNS WITH P1 AT 1RST NON-BLANK.
11400 FORP1=1 TO LEN(C$)
11500 IF C$(P1,P1)<>" "THENEXIT11800
11600 NEXT
11700 R=3\RETURN R
11800 IFLEN(C$)>1THEN11900\R=1\RETURN R
11900 C1$=C$(P1,P1+1)
12000 IF C1$="CY"THENR=2ELSE12100\GOTO12700
12100  IF C1$="TR"THENR=4ELSE12200\GOTO12700
12200   IF C1$="SA"THENR=5ELSE12300\GOTO12700
12300    IF C1$="LF"THENR=6ELSE12400\GOTO12700
12400     IF C1$="LC"THENR=7ELSE12500\GOTO12700
12500     IF C1$="TI"THENR=8ELSE12600\GOTO12700
12600     IF C1$="BY"THENR=9ELSER=1
12700 RETURN R
12800 FNEND
12900 REM ***********************************************
13000 DEF FNN(T)\REM DETERMINES IF P1 POINTING AT NUMBER.
13100 V=0\REM...ASSUME LOOKING AT A NON-NUMBER
13200 IFT<=LEN(C$)THEN13300\P1=4\RETURN V
13300 IFC$(T,T)<"0"THEN RETURN V
13400 IF C$(T,T)>"9"THENRETURN V
13500 V=1\RETURN V
13600 FNEND
13700 REM ***********************************************
13800 DEF FNV(T)\REM  RETURNS NUMERIC VALUE STARTING AT T
13900 FORP1=TTOLEN(C$)
14000 IF FNN(P1)=0THENEXIT14200
14100 NEXT
14200 V1=VAL(C$(T,P1-1))
14300 RETURN V1
```

```
14400 FNEND
14500 REM ************************************************
14600 REM  RETREIVES TIME OF DAY
14700 FOR I=1TO3\T4(I)=EXAM(T1+I)\NEXT
14800 RETURN
14900 REM ************************************************
15000 REM PRINTS MEMORY FROM A0 TO A1 W/S CHARACTERS/LINE
15100 DEF FNP(A0,A1,A2,S)
15200 REM  A0-START ADDRESS,A1-STOP ADDRESS,
15210 REM A2-OFFSET ADDRESS
15300 REM  S-CHARACTERS/LINE
15400 FORI=A0TOINT(A1/S)*S-1 STEP S
15500 PRINT#D,\PRINT#D,I
15600 FORJ=ITOI+S-1
15700 PRINT#D,%1I,CHR$(EXAM(A2+J)),\NEXTJ\NEXTI
15800 IFI>=A1THEN16100
15900 PRINT#D,\PRINT#D,I
16000 FORJ=ITOA1\PRINT#D,%1I,CHR$(EXAM(A2+J)),\NEXT
16100 PRINT#D,\PRINT#D,
16200 RETURN A1
16300 FNEND
16400 REM ************************************************
16500 REM XX,<SOURCE>,<DEST>(,<ADD>) ROUTINE
16600 IFP1+2>LEN(C$)THEN26000\P1=P1+2
16700 FORI3=0TO2
16800 IFP1<=LEN(C$)THEN17000ELSEIFI3<>2THEN26500
16900 A3=0\EXIT17800\REM NO<ADD>FOUND
17000 IFC$(P1,P1)<>","THENEXIT26000\P1=P1+1\P2=P1
17100 IFFNN(P1)=1THENEXIT26000
17200 T=FNB(P1)
17300 IFI3<>2THEN17500
17400 IFC$(P2,P1)<>"ADD"THENEXIT17900\A3=1\EXIT17800
17500 IFT=1THENEXIT26400\REM BLANK EMBEDDED
17600 IFI3=0THENS$=C$(P2,P1)ELSED$=C$(P2,P1)
17700 P1=P1+1\NEXT
17800 RETURN
17900 P1=P2\GOTO26000
18000 REM ************************************************
18100 REM TR,<SOURCE>,<DEST>(,<ADD>)
18200 GOSUB 16500
18300 IFFNC(S$)=0THEN26300\REM <SOURCE> D. N. E.
18400 F1=1\REM PREVENT CY FROM PRINTING
18500 PRINT#2,"B,B"\REM MAKES SURE NOT IN EDITOR
18600 GOSUB5300\REM WAIT FOR CYBER RESPONSE
18700 PRINT#2,"ATTACH,",D$,",ID=CYBER1"
18800 GOSUB5300
18900 FILLE1,1\REM SET EDIT FLAG FOR EDIT MODE
19000 PRINT#2,"EDITOR"
19100 GOSUB5300
19200 PRINT#2,"E,",D$
19300 GOSUB5300
19400 A1=FNA(C2)\REM ABS START OF CFILE
```

```
19500 A2=A1+FNA(C1)\REM ABS CURRENT POINTER
19600 C$="WARNING-EDIT FILE NOT SAVED"
19700 T=FNS(A1,A2,C$)
19800 IFT=1THEN19200
19900 IFA5=1THEN20100\REM ADD TO EXISTING FILE
20000 PRINT#2,"D,A"\REM DELETE FILE CONTENTS
20100 PRINT#2,"A,S,0"\REM TRANSFER NEW CONTENTS.
20200 T=FNF(S$,F2,2)\REM TRANSFER
20300 REM *********************************************/
20400 REM SCANS UNTIL A" "OR, IS FOUND.
20410 REM P1 POINTS TO CHARACTER
20500 REM JUST BEFORE " " OR ,.
20600 DEF FNB(T)
20700 V4=0\REM ASSUME COMMA
20800 FORT=P1TOLEN(C$)
20900 IFC$(T,T)=" "THENEXIT21100
20910 IFC$(T,T)=","THENEXIT21200
21000 NEXT\V4=2\P1=T-1\RETURN V4\REM NEITHER FOUND
21100 V4=1
21200 P1=T-1\RETURN V4
21300 FNEND
21400 REM *********************************************
21500 REM SEARCH CFILE FROM A1 TO A2
21510 REM FOR STRING C$. A1 RETURNED
21600 REM WITH\ 1-MATCH, 0-NO MATCH.
21700 DEF FNS(A1,A2,C$)
21800 IFA2-A1<LEN(C$)THEN22500\REM C$ TOO LONG
21900 FORI=A1TOA2-LEN(C$)-1
22000 FORJ=1TOLEN(C$)
22100  IFC$(J,J)<>CHR$(EXAM(I+J-1))THENEXIT22400
22200 NEXTJ
22300 A1=1\EXIT22600\REM C$ FOUND
22400 NEXT I
22500 A1=0\REM C$ NOT FOUND
22600 RETURN A1
22700 FNEND
22800 REM *********************************************
22900 REM PRINTS SEQUENTIAL FILE C$,F2 ON DEVICE D.
23000 DEF FNF(C$,F2,D)
23100 OPEN #F2,C$
23200 IF TYP(F2)=0THEN23800
23300 IF TYP(F2)=1THEN 23600
23400 READ #F2,T\PRINT#D,N,
23500 GOTO23200
23600 READ #F2,T$\PRINT#D,T$,
23700 GOTO23200
23800 RETURN F2
23900 CLOSE#F2
24000 FNEND
24100 REM *********************************************
24200 REM SA,<SOURCE>,<DEST>(,<ADD>)
24300 GOSUB 16500\REM PARSE SA
24400 IFFNC(D$)=0THEN26600\REM <DEST> D. N. E.
```

```
24500 PRINT#3,"FILES"\F1=1\GOSUB5300
24600 A(0)=FNA(C2)
24700 A(1)=FNA(C1)+A(0)
24800 IFFNS(A(0),A(1),S$)=0THEN26300
24900 PRINT#3,"REWIND,",S$\GOSUB5300
25000 PRINT#3,"COPYSBF,",S$,",OUTPUT"\F1=0\GOSUB5300
25010 A(0)=FNA(C2)\A(1)=FNA(C1)
25090 T=CALL(T6)
25100 OPEN#1,D$
25210 WRITE #1 %0,&EXAM(C1),&EXAM(C1+1),NOENDMARK
25300 FORI4=0TOA(1)\IFI4=M1THENEXIT25600
25400 WRITE #1 %I4+2,&EXAM(I4+A(0)),NOENDMARK
25500 NEXT
25510 L=FNA(C1)
25600 CLOSE#1
25610 T=CALL(T5)
25700 RETURN
25800 REM ************************************************
25900 T=FNE(1)\RETURN
26000 T=FNE(2)\RETURN
26100 T=FNE(3)\RETURN
26200 T=FNE(4)\RETURN
26300 T=FNE(5)\RETURN
26400 T=FNE(6)\RETURN
26500 T=FNE(7)\RETURN
26600 T=FNE(8)\RETURN
26700 T=FNE(9)\RETURN
26800 REM ************************************************
26900 REM CALCULATES ADDRESSES
27000 DEF FNA(T)
27100 T=EXAM(T)+EXAM(T+1)*256\RETURN T
27200 FNEND
27300 REM ************************************************
27400 REM PRINTS FILE C$,F2,ON DEVICE D FROM 0 TO A1.
27500 DEF FNL(C$,F2,D,A1)
27510 T=CALL(T6)\REM DI
27600 OPEN #F2,C$
27610 READ#F2 %0,&L,&L1\L=L+L1*256
27620 IF A1=99999THENA1=L
27700 FORI=0TOA1\IFI=M1THEN28100
27800 READ#F2 %I+2,&T
27900 PRINT#D,CHR$(T),
28000 NEXT
28100 CLOSE #F2
28110 T=CALL(T5)
28200 RETURN A1
28300 FNEND
28400 REM ************************************************
28500 REM DETERMINE IF C$ EXISTS.
28600 DEF FNC(C$)
```

```
28700 FORI=1TOLEN(C$)\FILL(C7+I-1,ASC(C$(I,I))\NEXT
28710 FILL(C7+I-1,32
28800 T=CALL(F)\IFEXAM(C6)=0THENT=0ELSET=1\RETURNT
28900 FNEND
29000 REM **************************
29001 REM LF,<SOURCE>,((<STOP>)(ALL))(,<DEVICE>)
29100 IFF1+2>LEN(C$)THEN26000\F1=F1+2
29200 FORI9=0TO2\IFF1<=LEN(C$)THEN29400ELSEIFI9<>2THENEXIT26500
29300 A(1)=1\EXIT30000\REM NO DEVICE
29400 IFC$(F1,F1)<>","THENEXIT26000\F1=F1+1\F2=F1
29500 IFI9<>0THEN29700\IFFNN(F1)=1THENEXIT26000\T=FNB(F1)
29600 IFT=1THENEXIT26400\S$=C$(F2,F1)\F1=F1+1\GOTO29900
29700 IFI9=2THEN29800\IFFNN(F1)<>0THEN29750
29705 IFF1+2>LEN(C$)THEN26000
29710 IFC$(F1,F1+2)="ALL"THENA(0)=99999ELSEEXIT26000\F1=F1+3
29720 GOTO29900
29750 A(0)=FNV(F1)\GOTO29900
29800 A(1)=FNV(F1)
29900 NEXT
29910 REM PRINT FILE
30000 IFFNC(S$)=0THEN26600\T=FNL(S$,1,A(1),A(0))
30100 RETURN
```

Appendix F

TIME Program Listing

```
        ORG     X'FE00'
TINIT   MVI     A,0360
        OUT     0376
        EI
        RET
TIME    PUSH    PSW
        PUSH    B
        PUSH    H
        LDA     CURLEV
        PUSH    PSW
        MVI     A,011
        STA     CURLEV
        ORI     0330
        OUT     0376
        EI
        MVI     B,02
        LXI     H,NMB
        MOV     A,M
        INR     M
        SUI     95
        JNZ     CNTLP
        MOV     M,A
        INX     H
LOOP    MOV     A,M
        INR     M
        SBI     59
        JNZ     CNTLP
        MOV     M,A
        INX     H
        DCR     B
        JNZ     LOOP
        MOV     A,M
        INR     M
        SBI     23
        JNZ     CNTLP
        MOV     M,A
CNTLP   MVI     B,02
        LXI     H,TICNT
        MOV     A,M
        INR     M
        SUI     95
        JNZ     CNTCK
        MOV     M,A
        INX     H
LOOP1   MOV     A,M
        INR     M
        SBI     59
        JNZ     CNTCK
        MOV     M,A
        INX     H
```

```
                DCR     B
                JNZ     LOOP1
                MOV     A,M
                INR     M
                SBI     23
                JNZ     CNTCK
                MOV     M,A         ;HOURS
        CNTCK   LXI     H,TICNT
                LDA     TISET       ;.01 SECS
                CMP     M
                JNZ     OUTLP
                INX     H
                LDA     TISET+1     ;SECS
                CMP     M
                JNZ     OUTLP
                INX     H
                LDA     TISET+2     ;MINS
                CMP     M
                JNZ     OUTLP
                INX     H
                LDA     TISET+3     ;HRS
                CMP     M
                JNZ     OUTLP
        MATCH   LXI     H,TIFLG
                MOV     A,M
                ANI     0377
                JZ      OUTLP
                ANI     01
                CNZ     TITRP1
                MOV     A,M
                ANI     02
                CNZ     TITRP2
                MOV     A,M
                ANI     04
                CNZ     TITRP3
                MOV     A,M
                ANI     010
                CNZ     TITRP4
                MOV     A,M
                ANI     020
                CNZ     TITRP5
                MOV     A,M
                ANI     040
                CNZ     TITRP6
                MOV     A,M
                ANI     0100
                CNZ     TITRP7
                MOV     A,M
                ANI     0200
                CNZ     TITRP8
```

```
OUTLP   DI
        POP     PSW
        STA     CURLEV
        ORI     0300
        OUT     0376
        POP     H
        POP     B
        POP     PSW
        EI
        RET
        EI
        RET
        DI
        RET
        ORG     000147
NMB     DS      5
        ORG     000154
CURLEV  DB      0
        ORG     0243
TICNT   DS      4
TISET   DS      4
TIFLG   DB      01
TITRP1  DS      3
TITRP2  DS      3
TITRP3  DS      3
TITRP4  DS      3
TITRP5  DS      3
TITRP6  DS      3
TITRP7  DS      3
TITRP8  DS      3
TIPRG1  ORG     X'2900'
        LXI     H,TI1MSG
        MVI     A,02
        CALL    MSG
        CALL    TIRST
        RET
TI1MSG  DB      "HELLO",015,0
        ORG     0176656
MSG     NOP
TIRST   ORG     0176757
        NOP
        END
```

Vita

Captain Donald L. Ravenscroft was born on 15 February 1952 in Yamagata, Japan. He graduated from high school in El Paso, Texas in 1970. He attended the United States Air Force Academy and graduated in 1974 with a Bachelor of Science degree and received a regular commission as a Second Lieutenant in the United States Air Force. After graduation he was assigned to the NAVSTAR Global Positioning System (GPS) Joint Program Office (JPO), Space and Missile Systems Organization, Los Angeles Air Force Station as a computer systems analyst. His job while assigned to the GPS JPO included software and hardware contract management and computer systems design. He entered the Air Force Institute of Technology School of Engineering in August 1977.

Permanent address: 1037 Lomita Dr.
El Paso, Texas 79907

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFIT/GCS/EE/78-16	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Electrical Engineering Digital Design Laboratory Communications Network		5. TYPE OF REPORT & PERIOD COVERED Master of Science Thesis
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Donald L. Ravenscroft, Captain, USAF		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Air Force Institute of Technology (AFIT/EN) Wright-Patterson AFB, Ohio 45433		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE December 1978
		13. NUMBER OF PAGES
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

Approved for public release; IAW AFR 190-17
Joseph P. Hipps, Maj, USAF [illegible] 1-23-79
Director of Information

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Distributed Network	Distributed Routing Algorithm
Network Hierarchy	Hold Down Technique
Network Architecture	Node-to-Node Protocol
General Routing Algorithm	

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Four types of network routing algorithms were investigated as candidates for use in the proposed AFIT Electrical Engineering Digital Engineering Laboratory (DEL) network. The four types were deterministic, isolated, centralized and distributed. The advantages and disadvantages of each type of network routing algorithm were evaluated for possible use in the DEL network. The distributed routing algorithm was selected for the proposed DEL network because it was found to be more efficient and reliable. The educational benefits of the distributed routing algorithm were also discussed. In order to improve the

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Block 20 Continued

distributed routing algorithm's response time to changes in the network topology and traffic flow and to reduce the algorithm's oscillation caused by changes in the network topology a technique known as "hold down" was incorporated. A description of the routing algorithm, including this "hold down" technique, is discussed in sufficient detail to permit implementation of the algorithm into the proposed DEL network.

In addition to the description of the distributed routing algorithm, several types of communications protocols were investigated for use in the node-to-node network. The Advanced Data Communications Control Procedures and the Synchronous Data Link Control procedures were recommended for use in the proposed DEL network.

Another subject investigated was the development and implementation of a data link between one of the nodes in the proposed DEL network and the CDC CYBER 74 computer. This data link provides the capability to send data files between a network node (an Altair 8800b computer) and the CYBER 74 computer. The data interface allows the user to selectively manipulate either the Altair computer software using the Altair operating system or the CYBER 74 system software using the CDC INTERCOM system. The selection of either system is easily accomplished using simple user commands. File transfers between the two computers is controlled using the interface software developed in this investigation. The ability to transfer files between the Altair computer and the CYBER 74 computer will allow the CYBER 74 computer to be used as a DEL network resource once the DEL network is developed.

The thesis is organized in three parts. Parts 1 and 2 describe the distributed routing algorithm and the Altair/CYBER 74 interface program respectively. Part 3 is the User's Manual for the Altair/CYBER 74 data interface program and is published under a separate cover.